Introducción a LabVIEW, uso de estructuras y funciones básicas
Tutorial de LabVIEW. Se explica el uso de las diversas Paletas, funciones básicas y se familiariza con el entorno en general. Este tutorial es usado en la cátedra Señales y Sistemas Continuos de la UCAB, y fue realizado por la profesora Maria Gabriela Rodriguez. El tutorial se realizó con la ayuda del "Curso de LabVIEW Seis Horas" de National Instruments, y con la ayuda del tutorial ubicado en "links"

LabVIEW es un entorno de programación destinado al desarrollo de aplicaciones, similar a los sistemas de desarrollo comerciales que utilizan el *lenguaje C* o *BASIC*. Sin embargo, LabVIEW se diferencia de dichos programas en un importante aspecto: los citados lenguajes de programación se basan en líneas de texto para crear el código fuente del programa, mientras que LabVIEW emplea la programación gráfica o *lenguaje G* para crear programas basados en diagramas de bloques.

Para el empleo de LabVIEW no se requiere gran experiencia en programación, ya que se emplean iconos, términos e ideas familiares a científicos e ingenieros, y se apoya sobre símbolos gráficos en lugar de lenguaje escrito para construir las aplicaciones. Por ello resulta mucho más intuitivo que el resto de lenguajes de programación convencionales. LabVIEW posee extensas librerías de funciones y subrutinas. Además de las funciones básicas de todo lenguaje de programación, LabVIEW incluye librerías específicas para la adquisición de datos, control de instrumentación VXI, GPIB y comunicación serie, análisis presentación y guardado de datos.

¿Cómo trabaja LabVIEW?

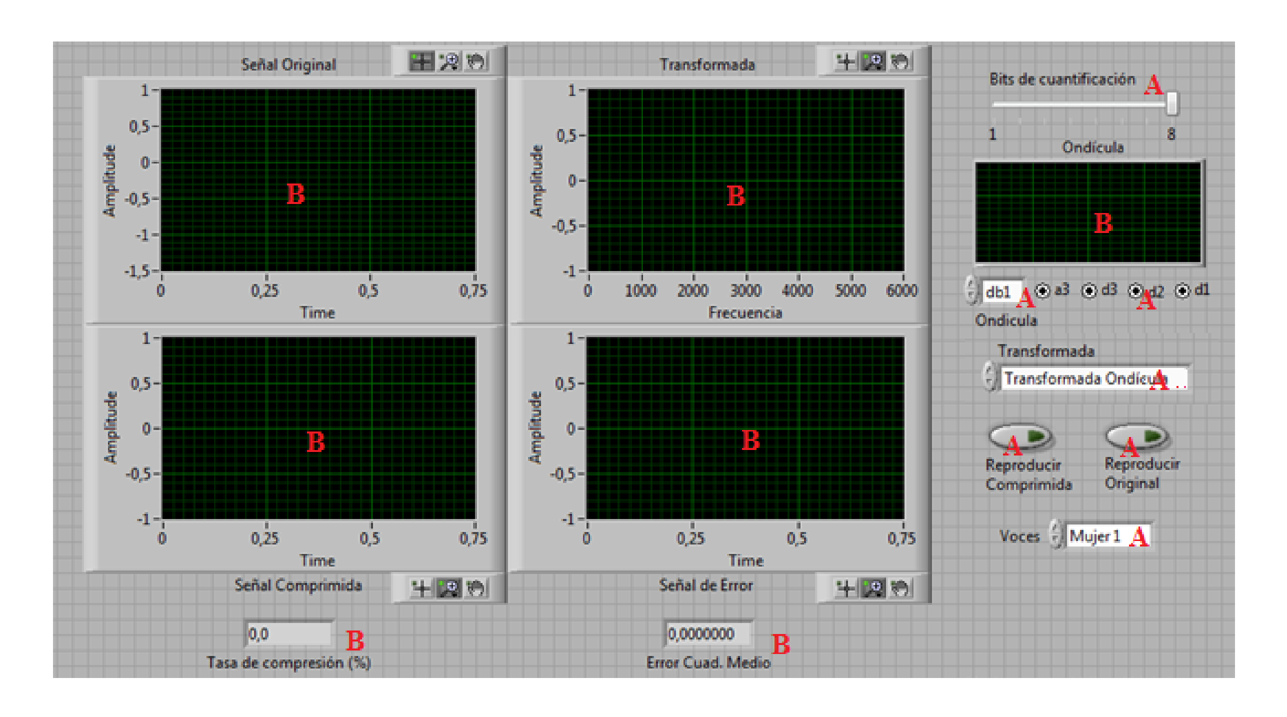
Los programas desarrollados mediante LabVIEW se denominan *Instrumentos Virtuales* (*VIs*), porque su apariencia y funcionamiento imitan los de un instrumento real. Sin embargo son análogos a las funciones creadas con los lenguajes de programación convencionales. Los *VIs* tienen una parte interactiva con el usuario y otra parte de código fuente, y aceptan parámetros procedentes de otros *VIs*.

Cada VI contiene tres partes principales:

- Panel frontal: Cómo el usuario interacciona con el VI.
- Diagrama de bloque: El código que controla el programa.
- Icono/Conector: Medios para conectar un VI con otros VIs.

Panel Frontal

Esta interfaz recoge las entradas procedentes del usuario y representa las salidas proporcionadas por el programa. Un *panel frontal* está formado por una serie de botones, pulsadores, potenciómetros, gráficos, etc. Cada uno de ellos puede estar definido como un *control* (a) o un *indicador* (b). Los primeros sirven para introducir parámetros al VI, mientras que los indicadores se emplean para mostrar los resultados producidos, ya sean datos adquiridos o resultados de alguna operación.



Panel frontal con controles (A) e indicadores (B)

Diagrama de bloques

El *diagrama de bloques* constituye el código fuente del *VI*. En el *diagrama de bloques* es donde se realiza la implementación del programa del *VI* para controlar o realizar cualquier procesado de las entradas y salidas que se crearon en el *panel frontal*.

El *diagrama de bloques* incluye *funciones* y *estructuras* integradas en las librerías que incorpora LabVIEW. En el *lenguaje G* las *funciones* y las *estructuras* son nodos elementales. Son análogas a los operadores o librerías de funciones de los lenguajes convencionales.

Los *controles* e *indicadores* que se colocaron previamente en el Panel Frontal, se materializan en el diagrama de bloques mediante los *terminales*

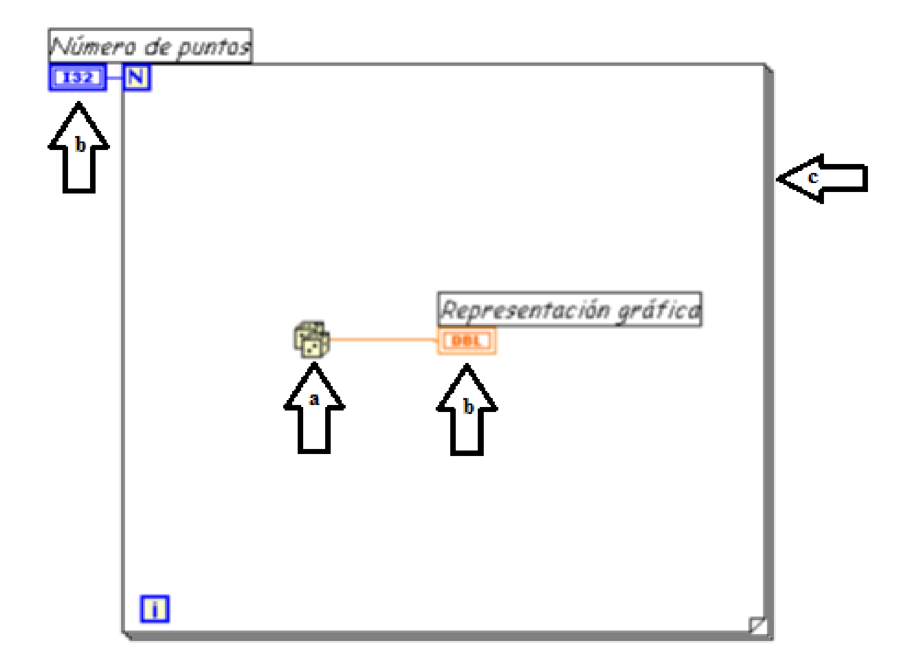


Diagrama de bloques con una función (a), dos terminales (control e indicador) (b) y una estructura (c)

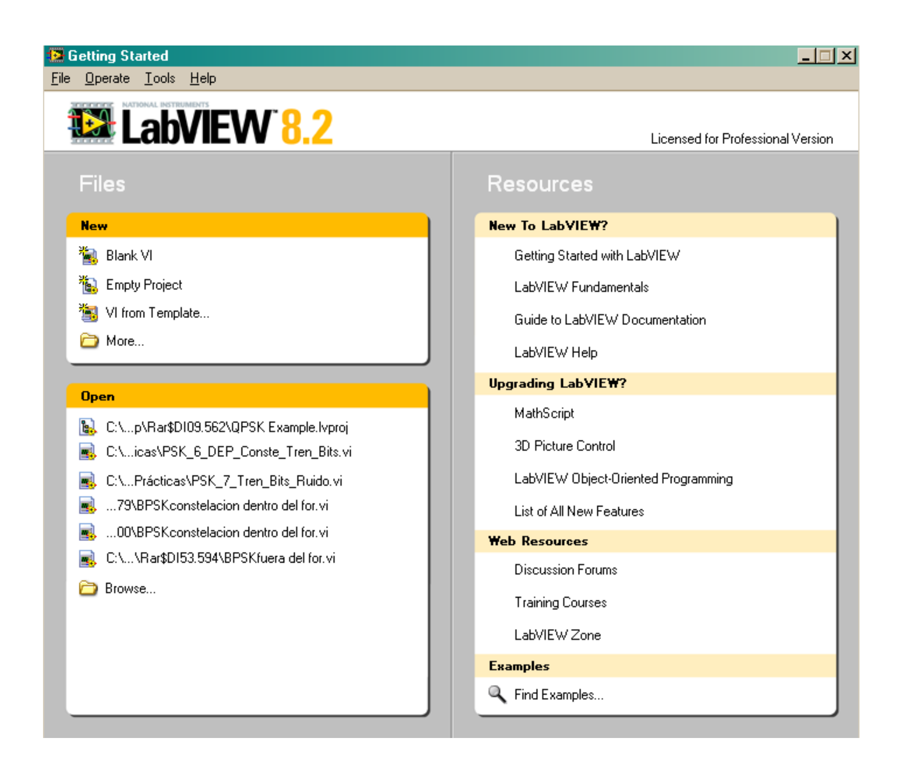
Como acceder al Programa:

Seleccione el botón **Inicio** → **Todos los Programas** → y localice el ejecutable del LabVIEW. Éste será similar al que se muestra a continuación:



Haga clic sobre el botón para iniciar el programa.

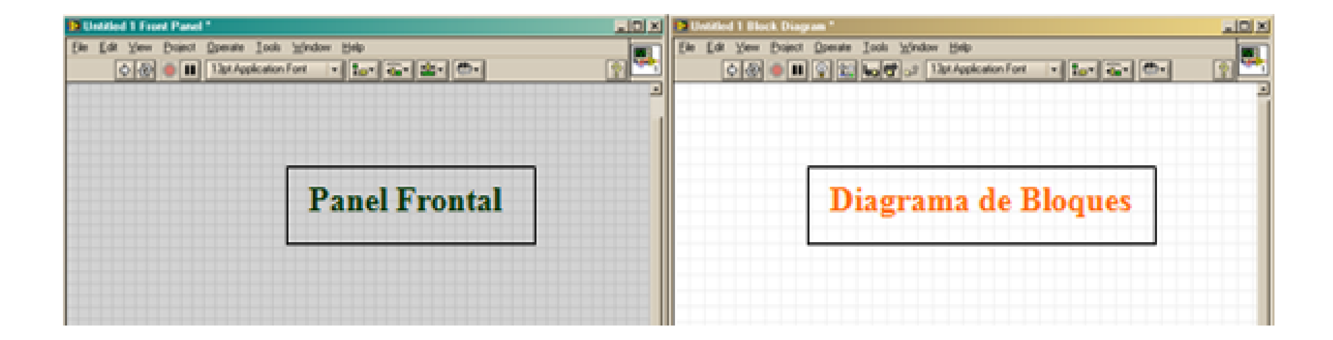
Una vez ejecutado el programa, aparecerá la siguiente pantalla:



Nuevo VI

Para entrar a las pantallas de programación, haga clic sobre el botón "**Blank VI**". Inmediatamente, aparecerán las pantallas del *Panel Frontal* y del *Diagrama de Bloque* en cascada; presione la combinación de teclas **ctrl.+T** y ambas pantallas se colocaran una al lado de la otra como se observa en la

figura 4. Revise el menú desplegable del botón *Window* de la barra de tareas de cualquiera de las pantallas para cambiar la disposición de las pantallas en su monitor.



Pantallas de LabVIEW

El **panel frontal** es la interfaz del usuario con el VI. El panel frontal se construye con controles e indicadores, que son las entradas y salidas que interactúan con las terminales del VI, respectivamente. Los controles son botones, botones de empuje, marcadores y otros componentes de entradas. Los indicadores son las graficas, luces y otros dispositivos. Los controles simulan instrumentos de entradas de equipos y suministra datos al diagrama de bloques del VI. Los indicadores simulan salidas de instrumentos y suministra datos que el diagrama de bloques adquiere o genera.

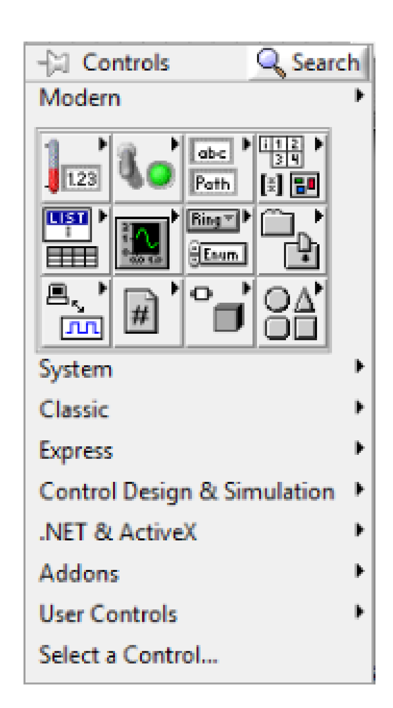
El **diagrama de bloques** contiene el código fuente grafico. Los objetos del panel frontal aparecen como terminales en el diagrama de bloques. Adicionalmente, el diagrama de bloques contiene funciones y estructuras incorporadas en las bibliotecas de LabVIEW VI. Los cables conectan cada uno de los nodos en el diagrama de bloques, incluyendo controles e indicadores de terminal, funciones y estructuras.

Paletas

Las paletas de LabVIEW proporcionan las herramientas que se requieren para crear y modificar tanto el panel frontal como el diagrama de bloques. Existen las siguientes paletas:

Paleta de controles

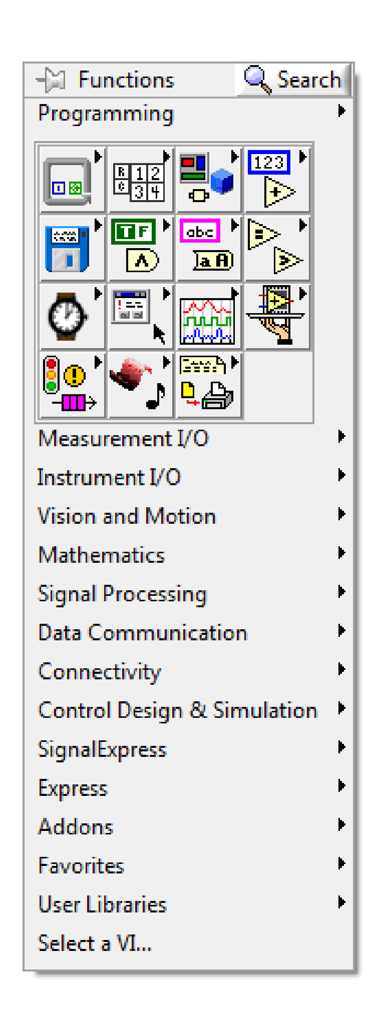
La **paleta de controles** (*Controls*) se usa para colocar los controles e indicadores en el panel frontal. La paleta de controles esta disponible solamente en el panel frontal. Seleccionando *View → Controls palette* o haciendo clic derecho en el espacio de trabajo en el panel frontal se despliega esta paleta. También puede desplegarse la paleta de controles haciendo un clic derecho en un área abierta del panel frontal. Para desaparecer esta paleta se hace clic izquierdo en cualquier área abierta del panel



Controls Palette

Paleta de funciones

Se usa la **paleta de funciones** (*Functions*), para construir un diagrama de bloques. La paleta de funciones esta disponible solamente en el diagrama de bloques. Seleccionando *View → Functions Palette* o haciendo clic derecho en el espacio de trabajo del diagrama de bloques se despliega esta paleta. También puede desplegarse la paleta de funciones haciendo un clic derecho en un área abierta del diagrama de bloques. Para desaparecer la paleta se hace clic izquierdo en cualquier área abierta del panel



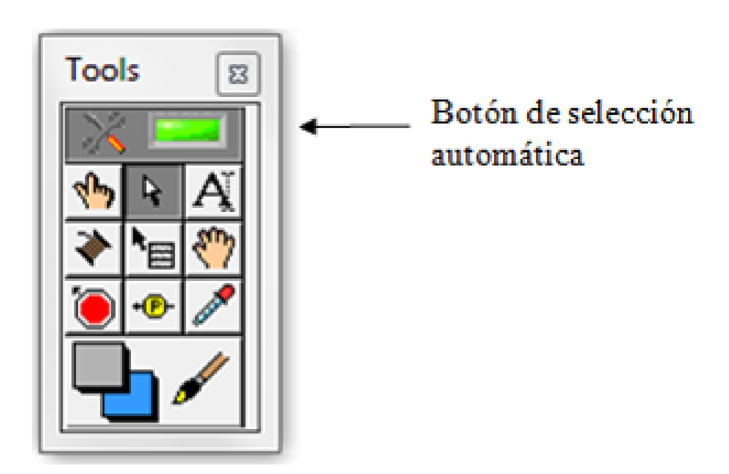
Functions Palette

Dependiendo de la versión de LabVIEW de la que se disponga, se tendrá un mayor o menor número de controles y funciones disponibles.

Se puede cambiar el formato de presentación de cada una de las paletas, para ello se selecciona el botón *View → View This Palette As* en cada una de las paletas y se escoje la opción que le sea más cómoda para trabajar. Se Recomienda explorar el resto de los botones que se encuentran dentro del botón *View* para ajustar cada paleta como resulte más cómodo.

Paleta de Herramientas

Esta paleta puede ser accedida desde cualquiera de las áreas de trabajo, seleccionando *View → Tools Palette*. Dispone de un botón de selección automática, si se encuentra seleccionado y se mueve el cursor sobre un objeto en el panel frontal o en el diagrama de bloque, LabVIEW automáticamente selecciona la herramienta correspondiente de la paleta de controles. En caso contrario, se deberá hacer la selección apropiada manualmente.



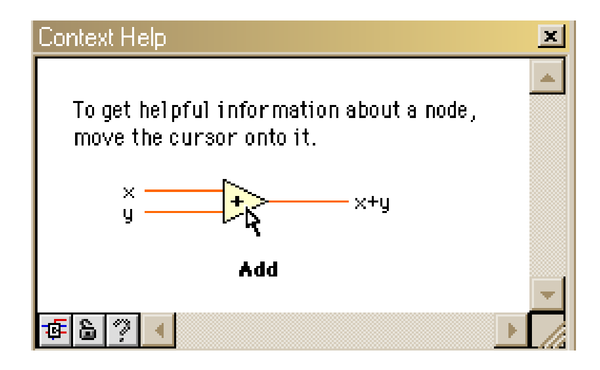
Tools Palette

Para mayor comodidad, asegúrese que se encuentre seleccionado.

La "Ayuda" de LabVIEW.

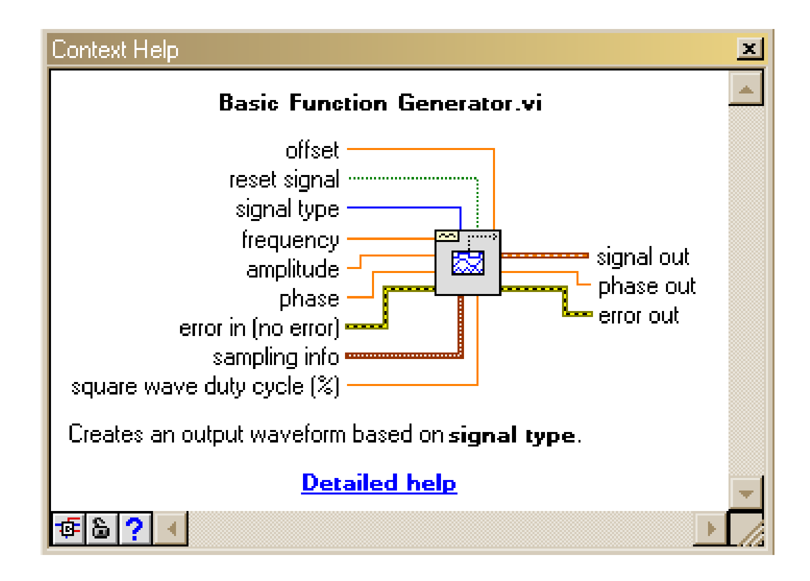
Como en cualquier otro programa, es muy importante obtener información de cómo operan las funciones y cuál es la sintaxis que debe seguirse para la programación.

Existen dos maneras básicas de obtener ayuda del programa, la primera es haciendo clic en *Help → Show Context Help*, a lo cual aparecerá la siguiente ventana:



Ayuda Contextual

Cuando se pase el cursor sobre cualquier VI, el contenido de la ventana *Context Help* cambiará y dará una ayuda rápida acerca del instrumento virtual sobre el cual esté el cursor.



Ayuda Contextual para un elemento

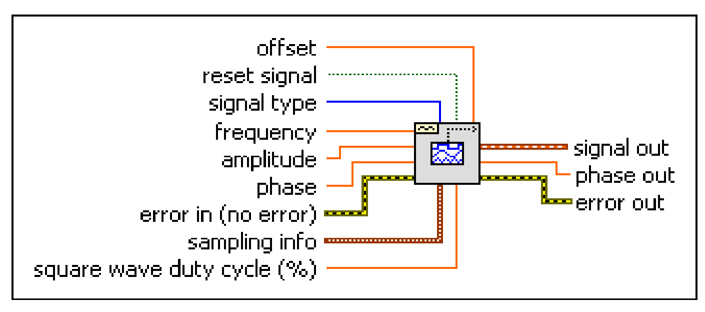
La otra manera de tener acceso a la ayuda es haciendo clic derecho sobre el VI del cual quiere obtener información; se desplegará una ventada en la cual debe seleccionar en botón que corresponde a *Help*. Casi de inmediato se abrirá una nueva ventana donde aparecerá de maneta detallada la información relacionada con el VI que está buscando junto con el nombre y utilidad de cada uno de los terminales de los que dispone el instrumento.

Tipos de Datos que emplea LabVIEW

Al igual que en otros lenguajes de programación, debe tenerse cuidado con el tipo de dato con el que se está trabajando; es decir, de acuerdo con el control o VI con el que se trabaje, éste podrá operar con datos del tipo: *Boolean*, *single*, *double*, binarios del tipo *byte*, *Word*, etc. LabVIEW hace la distinción asigna un color y un tipo de "cable" a cada estructura de datos; así éstos se pueden ver de la siguiente manera:

Basic Function Generator (Not in Base Package)

Creates an output waveform based on **signal type.** Details Example



Nótense los diferentes tipos de cable

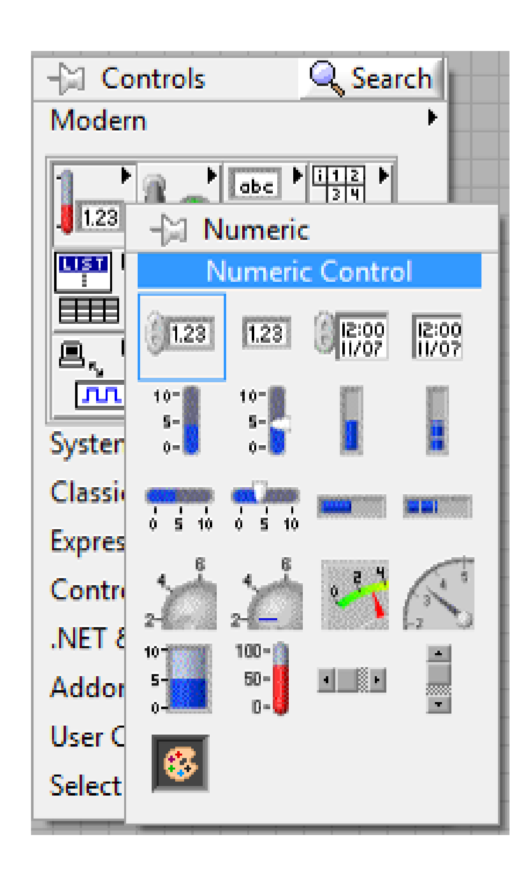
Por ejemplo, la línea delgada de color naranja representa datos de un solo tipo con valores decimales en los cuales se pueden encontrar los datos de precisión simple, doble o extendida. Las líneas delgadas de color azul son datos con o sin signo tipo *byte*, *Word* o *Long*. Las líneas más gruesas representan datos compuestos a los que se les llama *Cluster*, estos pueden estar compuestos con datos de diferente índole, para lo cual el programa se encarga de mantenerlos separados y organizados.

Una ventaja que ofrece LabVIEW es que al hacer conexiones entre VI con datos diferentes, en la mayoría de los casos, el programa se encarga de hacer la adaptación del tipo de dato simplificando, al usuario, la operación de conversión entre ellos. Por el contrario, si al programa se le hace imposible realizar la adaptación de los datos, entonces se presentará la conexión como un cable interrumpido.

Conexiones entre los diferentes controles e instrumentos

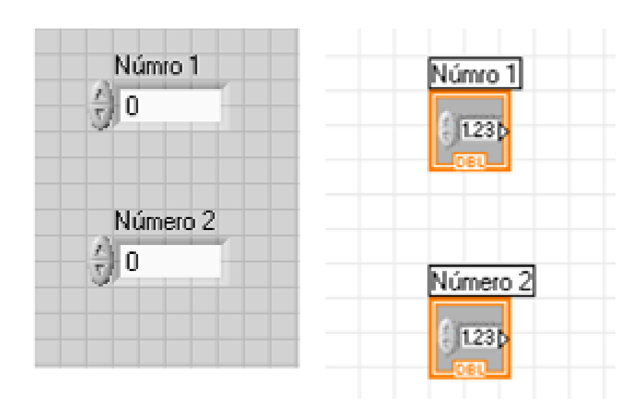
En la sección de Paleta de Herramientas, se indicó la conveniencia de mantener habilitado el botón de selección automática, esto permitirá ahorrar tiempo a la hora de manipular y hacer conexiones entre VI y/o controles.

En la pantalla del **Panel Frontal**, desde la Paleta de Controles haga clic hasta conseguir un control del tipo **Numeric Control**: *Modern → Numeric*



Numeric control. No debe mantener presionado el botón del *Mouse* para ubicarlo, sólo presione una vez y libere el botón.

El cursor, que antes era tipo puntero, ahora será tipo mano. Lleve el cursor hasta el panel frontal y haga clic en el sitio que desea colocar el control numérico que acaba de seleccionar (podrá colocar el nombre que desee a este control en este momento o en cualquier momento en el futuro).



Controles numéricos en el Panel de Control y en el Diagrama de Bloques

A la izquierda se muestra la manera en la que se verán los controles numéricos en el panel frontal. A la derecha se muestra la contraparte de los mismos controles que aparecerán simultáneamente en el diagrama de bloques. **Las conexiones sólo podrán realizarse en el diagrama de bloques.**

Para familiarizarse con las propiedades de estos controles, haga clic derecho sobre cualquiera de los controles numéricos y selecciones *Properties*. Explore las funciones de cada una de las pestañas de la ventana de propiedades.

Realice el mismo procedimiento para colocar, esta vez, indicadores numéricos, los mismos se encuentran en *Modern → Numeric → Numeric Indicator*. Coloque dos, uno con el nombre de **Suma** y el otro con el nombre de **Resta**.

Ahora seleccione la pantalla del Diagrama de Bloques. En la Paleta de Funciones seleccione *Programming → Numeric → Add*, arrastre el sumador hasta el lugar en la pantalla del diagrama de bloque donde desea colocarlo y haga clic para depositarlo. Consulte la Ayuda para revisar su funcionamiento.

Realice la misma operación pero esta vez para colocar un restador. Consulte la Ayuda para revisar su funcionamiento.

Una vez hecho todo lo anterior, deberá tenerse la siguiente programación en la pantalla de diagrama de bloques:

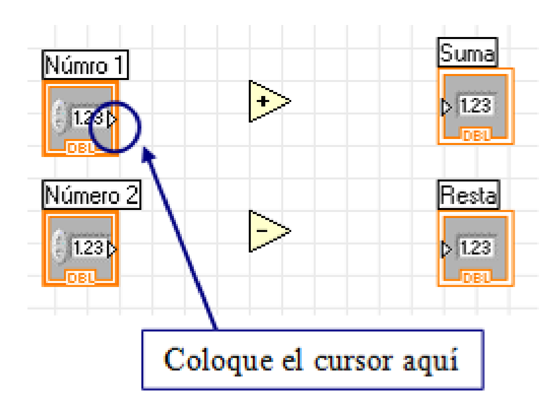
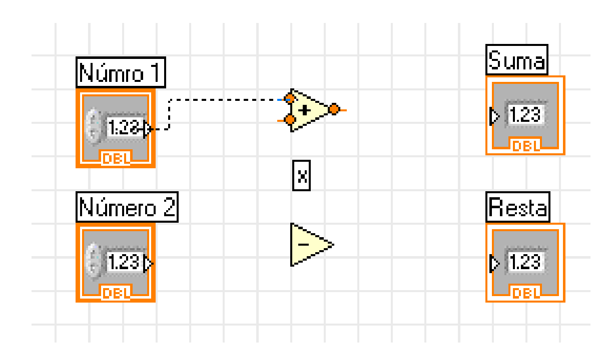


Diagrama de bloques sin cablear

Para realizar las conexiones coloque el cursor sobre el triángulo e inmediatamente éste cambiará a la forma de herramienta de cableado; haga clic sobre el terminal y mueva el cursor hasta uno de los terminales de la izquierda del sumador.



Realizando el cableado

Haga lo mismo con los demás terminales hasta obtener un resultado parecido al siguiente:

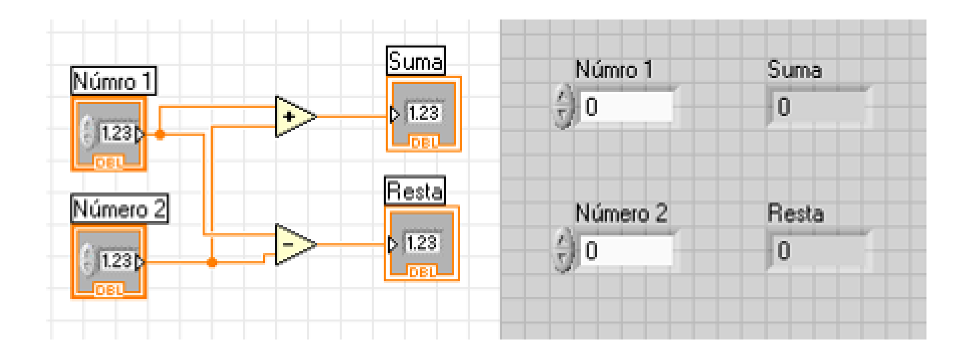


Diagrama de Bloques hechas las conexiones y Panel Frontal

Cómo correr el programa

Este proceso se logra a través de la Barra de Herramientas de Estados, la cual esta disponible desde cualquiera de las dos pantallas del programa.



Vista desde el Panel frontal



Vista desde el Diagrama de Bloques

	Descripción
	Botón de ejecución.
	Botón de ejecución continua.
	Cancelación de la ejecución.
	Botón de pausa/continuación.
	Botón de ejecución resaltada. Se emplea para depuración de errores

Botones en la barra de herramientas

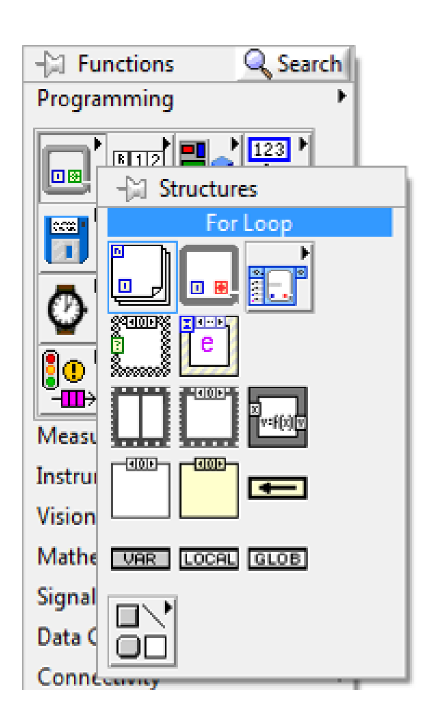
Desde el Panel Frontal, asigne valores diferentes a los controles numéricos haciendo clic dentro del control y escribiendo un número a través del teclado. Presione el botón de ejecución *Run* y observe el resultado en los indicadores.

Panel frontal de ejemple anterior

Pruebe cada una de las modalidades de ejecución y observe los resultados. Para el caso de ejecución continua, podrán cambiarse los valores de los números de manera dinámica y los resultados cambiaran de forma inmediata. Otra manera de cambiar el valor del control numérico es colocando el cursor sobre la parte izquierda, donde se encuentran las flechas, el cursor cambiará a tipo mano, haciendo clic sobre alguna de las flechas el valor del número se incrementará o disminuirá dependiendo del caso.

Estructuras

En la paleta de funciones la primera opción es la de las estructuras. Éstas controlan el flujo del programa, bien sea mediante la secuenciación de acciones, ejecución de bucles, etc.



Estructuras

Las estructuras se comportan como cualquier otro nodo en el diagrama de bloques, ejecutando automáticamente lo que está programado en su interior una vez tiene disponibles los datos de entrada, y una vez ejecutadas las instrucciones requeridas, suministran los correspondientes valores a los cables unidos a sus salidas. Sin embargo, cada estructura ejecuta su subdiagrama de acuerdo con las reglas específicas que rigen su comportamiento, y que se especifican a continuación.

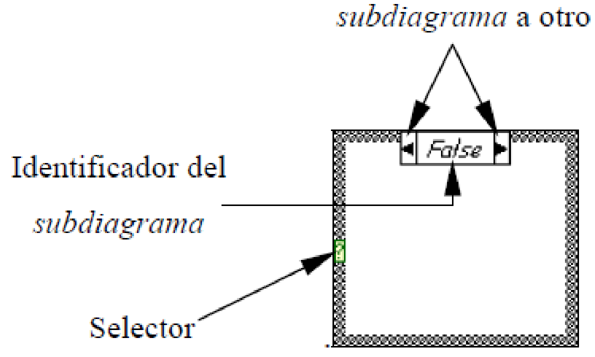
Un subdiagrama es una colección de nodos, cables y terminales situados en el interior del rectángulo que constituye la estructura. El *For Loop* y el *While Loop* únicamente tienen un subdiagrama. El *Case Structure* yel *Stacked Sequence Structure*, sin embargo, pueden tener múltiples subdiagramas, superpuestos como si se tratara de cartas en una baraja, por lo que en el diagrama de bloques únicamente será posible visualizar al tiempo uno de ellos. El *Flat Sequence Structure* posee varios subdiagramas colocados unos al lado de otros. Pueden agregarse más subdiagramas para las estructuras que lo permitan pulsando el botón derecho sobre el borde de la estructura y seleccionando la opción *Add Frame After* o *Add Frame Before* según donde se desee el nuevo subdiagrama. Los subdiagramas se construyen del mismo modo que el resto del programa

Las siguientes estructuras se hallan disponibles en el lenguaje G:

Case Structure

Al igual que otras estructuras posee varios subdiagramas, que se superponen como si de una baraja de cartas se tratara. En la parte superior del subdiagrama aparece el identificador del que se está representando en pantalla. A ambos lados de este identificador aparecen unas flechas que permiten pasar de un subdiagramaa otro.

En este caso el identificador es un valor que selecciona el subdiagrama que se debe ejecutar en cada momento.



Case Structure

La estructura *Case* tiene al menos dos subdiagramas(*True* y *False*). Únicamente se ejecutará el contenido de uno de ellos, dependiendo del valor de lo que se conecte al selector.

Stacked Sequence Structure

De nuevo, este tipo de estructuras presenta varios subdiagramas, superpuestos como en una baraja de cartas, de modo que únicamente se puede visualizar una en pantalla. También poseen un identificador del subdiagramamostrado en su parte superior, con posibilidad de avanzar o retroceder a otros subdiagramasgracias a las flechas situadas a ambos lados del mismo.

Botones para pasar de un *subdiagrama* a otro

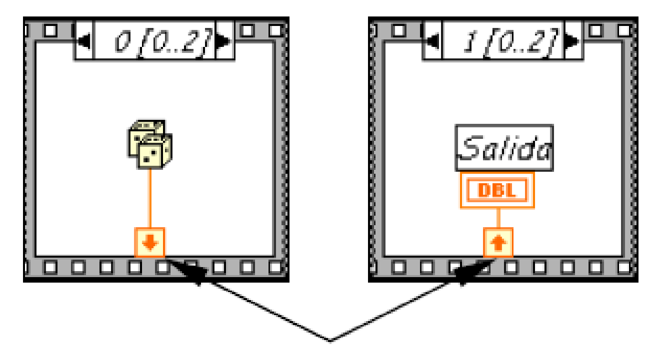
Identificador del *subdiagrama*

0 [0..2]

Stacked Sequence Structure

Esta estructura secuencia la ejecución del programa. Primero ejecutará el subdiagramade la hoja (*frame*) nº0, después el de la nº 1, y así sucesivamente.

Para pasar datos de una hoja a otra se pulsará el botón derecho del ratón sobre el borde de la estructura, seleccionando la opción *Add sequence local*.

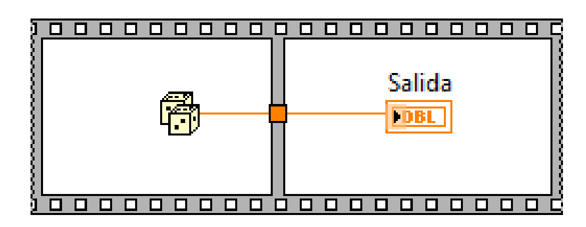


Sequence local: paso de un dato de la *frame 0* a la *1*

Datos de subdiagrama a subdiagrama en Stacked Sequence Structure

Flat Sequence Structure

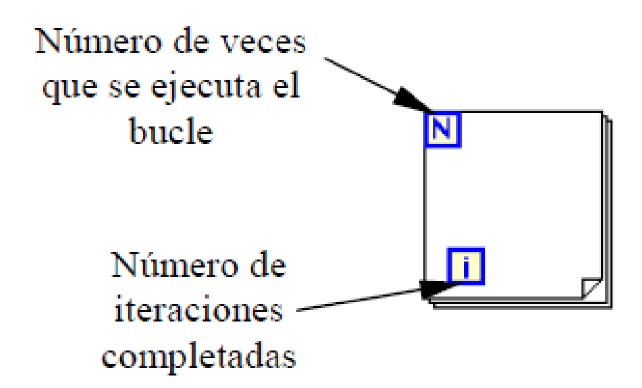
Su funcionamiento es similar al de la *Stacked Sequence Structure*, esta estructura tiene varios subdiagramas colocados uno al lado de otro, su orden de ejecución es de izquierda a derecha.



Flat Sequence Structure

For Loop

Es el equivalente al bucle *for* en los lenguajes de programación convencionales. Ejecuta el código dispuesto en su interior un número determinado de veces.



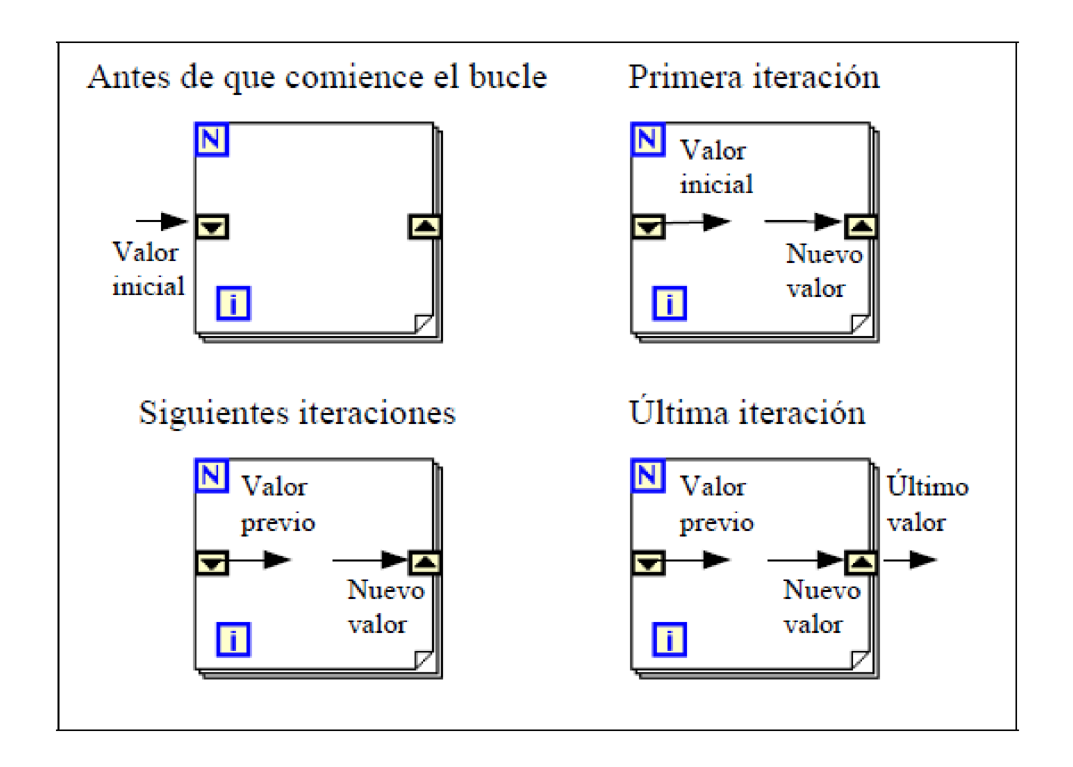
For Loop

Ejecutar el bucle *for* es equivalente al siguiente fragmento de código:

```
For i = 0 to N – 1
Ejecutar el subdiagrama del interior del Bucle
```

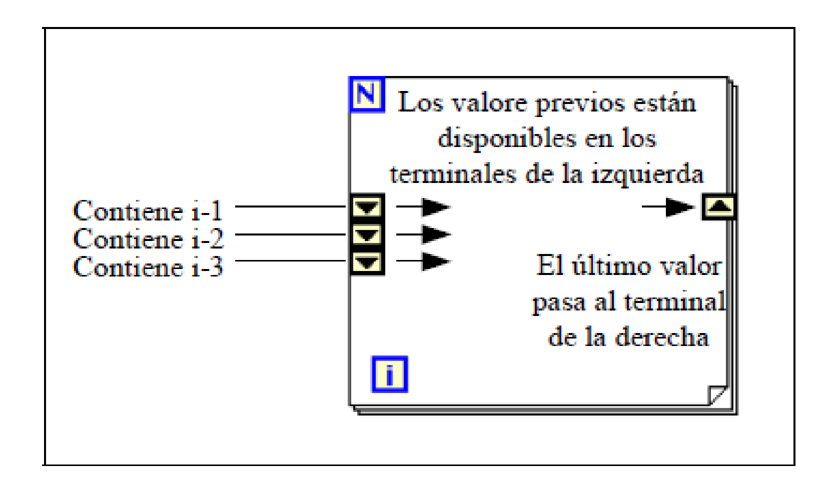
Código de For Loop

Para pasar valores de una iteración a otra se emplean los llamador *shift registers*. Para crear uno, se pulsará el botón derecho del ratón mientras éste se halla situado sobre el borde del bucle, seleccionando la opción *Add Shift Register*. El *shift register* consta de dos terminales, situados en los bordes laterales del bloque. El terminal izquierdo almacena el valor obtenido en la iteración anterior. El terminal derecho guardará el dato correspondiente a la iteración en ejecución. Dicho dato aparecerá, por tanto, en el terminal izquierdo durante la iteración posterior.



Datos de iteración a iteración en For Loop

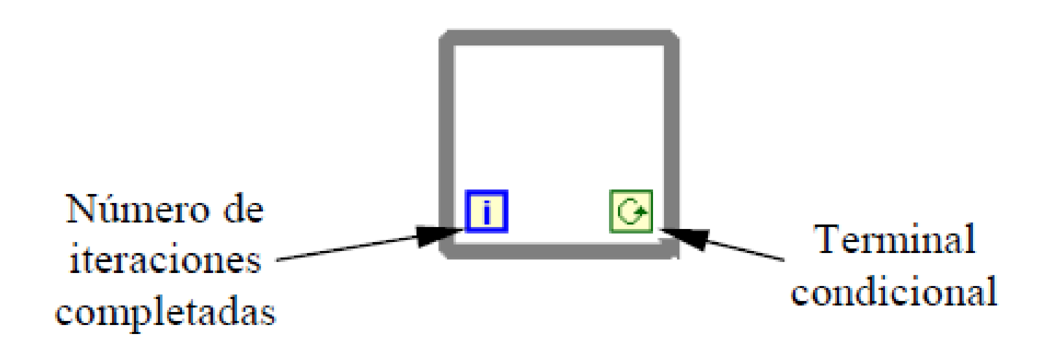
Se puede configurar un *shift register* para memorizar valores de varias iteraciones previas. Para ello, con el ratón situado sobre el terminal izquierdo del *shift register* se pulsará el botón derecho, seleccionando a continuación la opción *Add Element*.



Valores previos de variables en For Loop

While Loop

Es el equivalente al bucle *while* empleado en los lenguajes convencionales de programación. Su funcionamiento es similar al del bucle *for.*



Bucle While (While Loop)

El bucle *while* es equivalente al código siguiente:

Do *Se ejecuta lo que hay en el interior del bloque* *while terminal condicional is true*

Código de While Loop

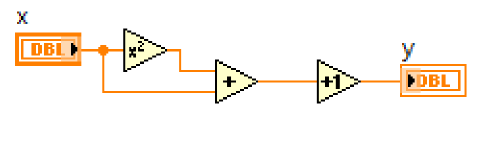
El programa comprueba el valor de lo que se halle conectado al terminal condicional al finalizar el bucle. Por lo tanto, el bucle siempre se ejecuta al menos una vez.

Con esta estructura también se pueden emplear los *shift registers* para tener disponibles los datos obtenidos en iteraciones anteriores (es decir, para memorizar valores obtenidos). Su empleo es análogo al de los bucles *for*, por lo que se omite su explicación.

Formula Node

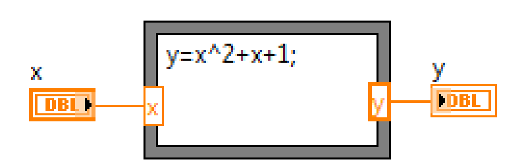
La estructura denominada *Formula Node* se emplea para introducir en el diagrama de bloques fórmulas de un modo directo. Resulta de gran utilidad cuando la ecuación tiene muchas variables o es relativamente compleja. Por ejemplo, se desea implementar la ecuación: *y = x2 + x + 1*

Empleando bloques pertenecientes al *lenguaje G* quedaría:



Ecuación con lenguaje G

Si se utiliza *formula node*, se obtiene:



Ecuación con Formula Node

Para definir una fórmula mediante este estructura, se actuará del siguiente modo:

- En primer lugar, se deben definir las variables de entrada y las de salida. Para ello, se pulsa con el botón derecho del ratón sobre el borde de la *formula node*. A continuación se seleccionará *Add Input* o *Add Output*, según se trate de una entrada o una salida, respectivamente. Aparecerá un rectángulo, en el que se debe escribir el nombre de la variable (se distingue entre mayúsculas y minúsculas). Todas las variables que se empleen deben estar declaradas como entradas o salidas. Las que se empleen como variables intermedias se declararán como salidas, aunque posteriormente no se unan a ningún bloque posterior.
- Una vez definidas las variables a emplear, se escribirán la o las fórmulas en el interior del recuadro. Cada fórmula debe finalizar con un “;”.
- Los operadores y funciones que se pueden emplear se explican en la ayuda de LabVIEW, y son los que se muestran a continuación:

Operadores:

asignación =

condicional ?:

OR lógico ||

AND lógico &&

relacionales == != > < >= <=

aritméticos + - * / ^

Funciones:

abs acos acosh asin asinh atan atanh ceil cos cosh cot csc exp expm1 floor getexp getman int intrz ln lnp1 log log2 max min mod rand rem sec sgn sin sinc sinh sqrt tan tanh

Operadores aplicables en Formula Node

Elaboración de un programa con LabVIEW

En secciones anteriores se hizo un pequeño ejemplo de un programa que suma y resta dos números. Para la siguiente parte se construirá un Generador de Funciones con un Osciloscopio para visualizar la señal generada y para finalizar un Analizador de Espectro.

Para acercarnos los más posible a la realidad, se incluirá en este programa un botón de encendido y apagado, con lo cual se ilustrará el concepto de estructuras (ciclo *Case*).

Para el Generador de Funciones se requiere:

Un VI de nombre: **Basic Funtion Generator.vi**, el cual será ubicado desde la Paleta de Funciones del Diagrama de Bloque.

En *Programming*→ *Waveform* → *Analog Waveform* → *Waveform Generation* Seleccione el VI **Basic FuncGen** y arrástrelo hasta la pantalla.

Coloque el cursor sobre el Generador que acaba de crear y explore los nombres de todos los terminales disponibles. Para poner a funcionar este instrumento no será necesario conectar todos los terminales.

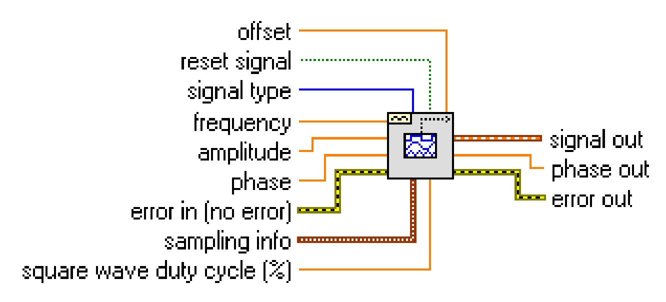
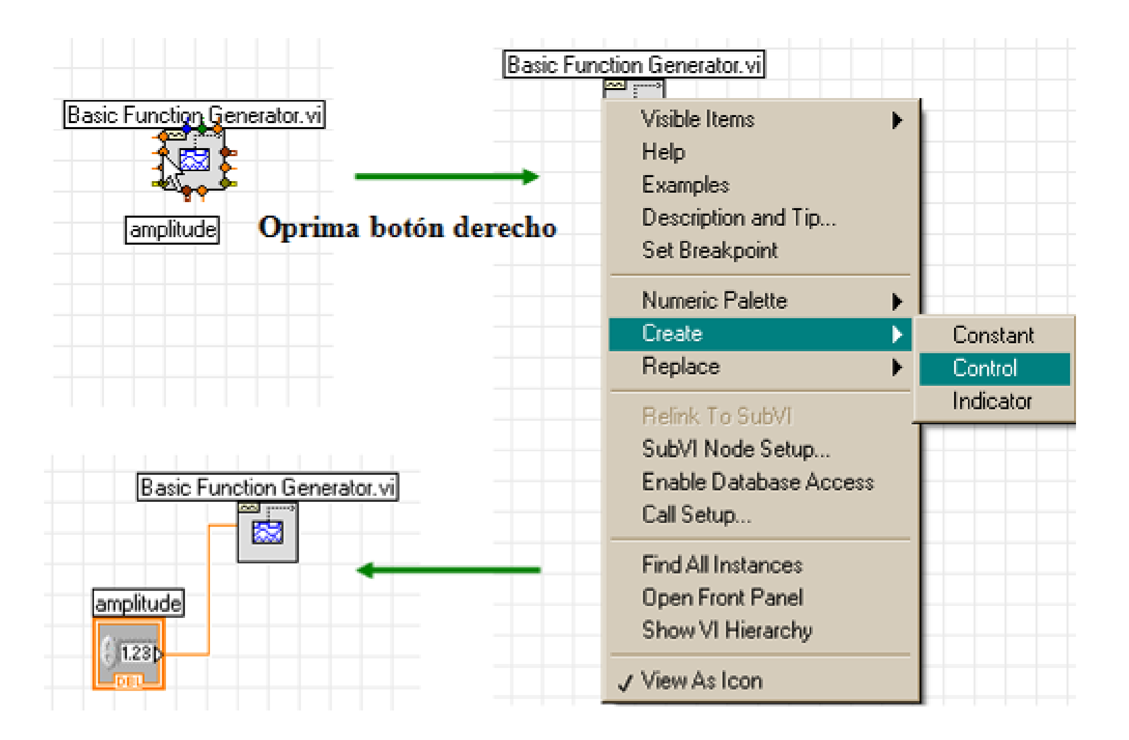


Imagen del Generador de Funciones vista desde la "Ayuda Contextual"

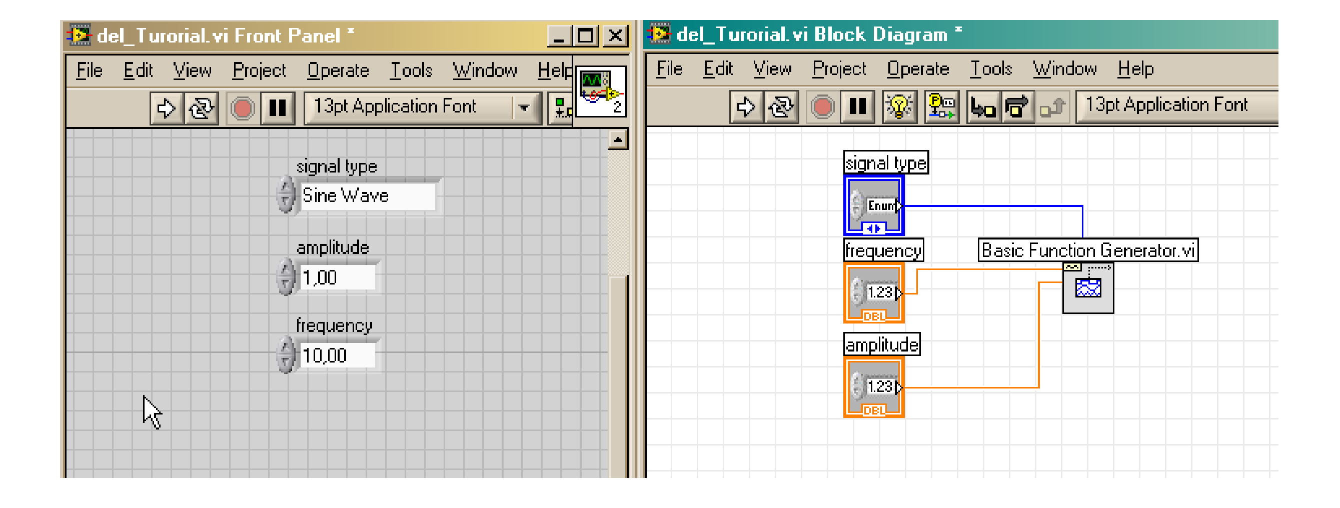
En un Generador de Funciones real, pueden controlarse cosas básicas como el tipo de señal, la amplitud o la frecuencia; y obtenerse cosas como la señal generada o de salida.

Para crear los controles de una manera rápida y sin riesgos de cometer errores haga clic derecho sobre el terminal *amplitude* del **Basic Function Genrator.vi** y seleccione *Create→Control*

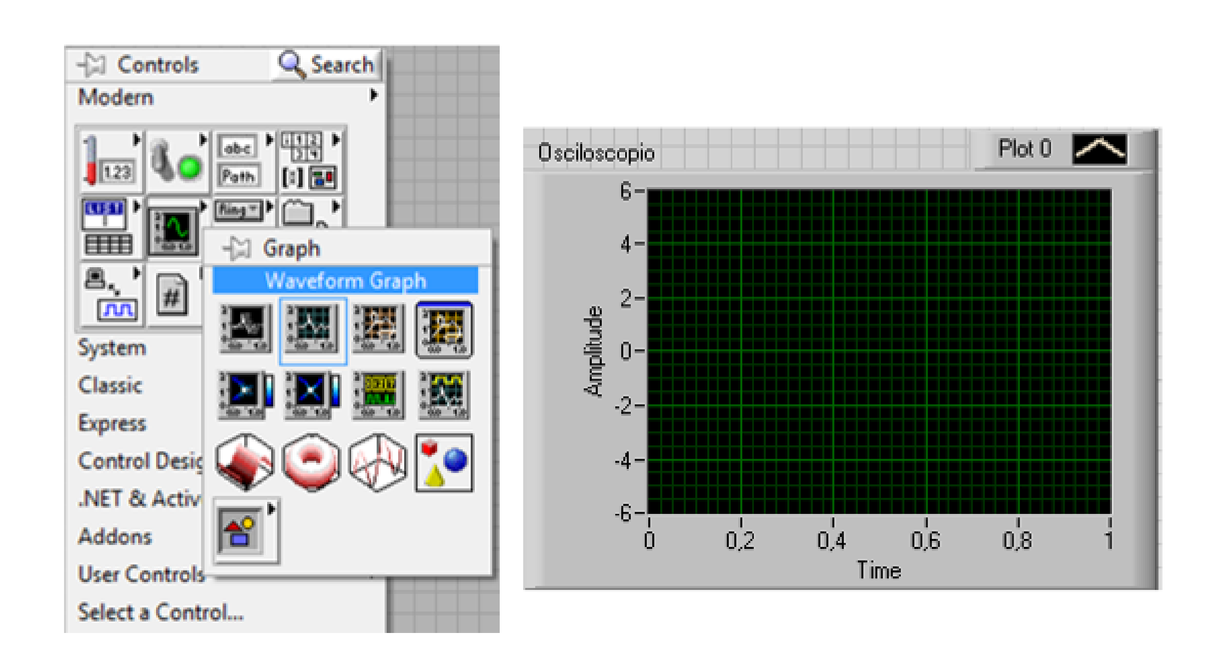


Método alternativo para crear un control

Realice el mismo procedimiento para crear el control de Frecuencia y el selector para el Tipo de Señal. Se deben crear de manera automática los correspondientes controles en el Panel Frontal.



Para colocar el Osciloscopio seleccione desde la Paleta de Controles en el Panel Frontal *Modern → Graph*, seleccione y arrastre el indicador gráfico **Wareform Graph**, coloque el nombre de Osciloscopio.



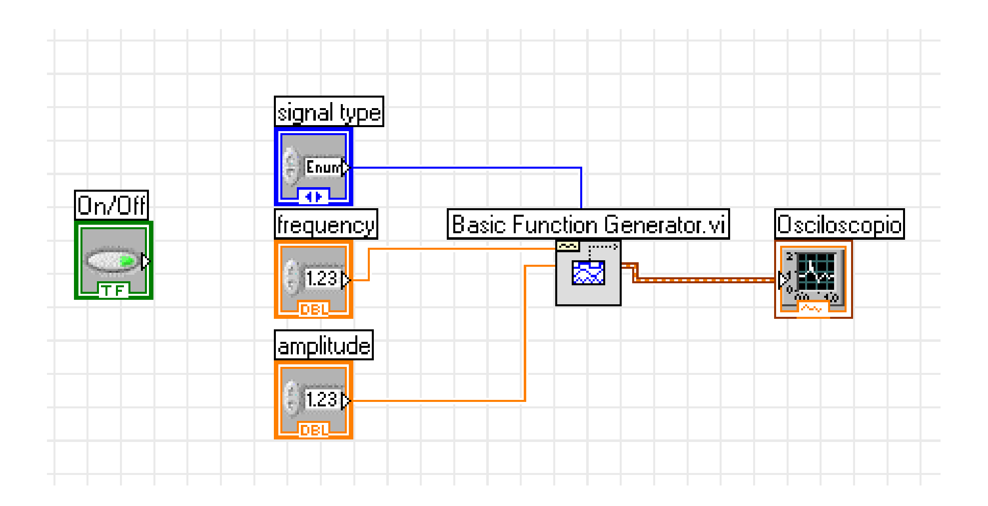
Localización de Waveform Graph

Revise la lista de operaciones que puede realizar haciendo clic botón derecho sobre el indicador gráfico que acaba de crear (tendrá diferentes opciones dependiendo el sitio donde ubique el cursor). Revise específicamente las etiqueta *Visible Ítems*, *X Scale*, *Y Scale* y *Properties*. También pruebe hacer clic botón derecho en la parte superior derecho del indicador, en el área de nombre **Plot 0**, explore todos los atributos disponibles.

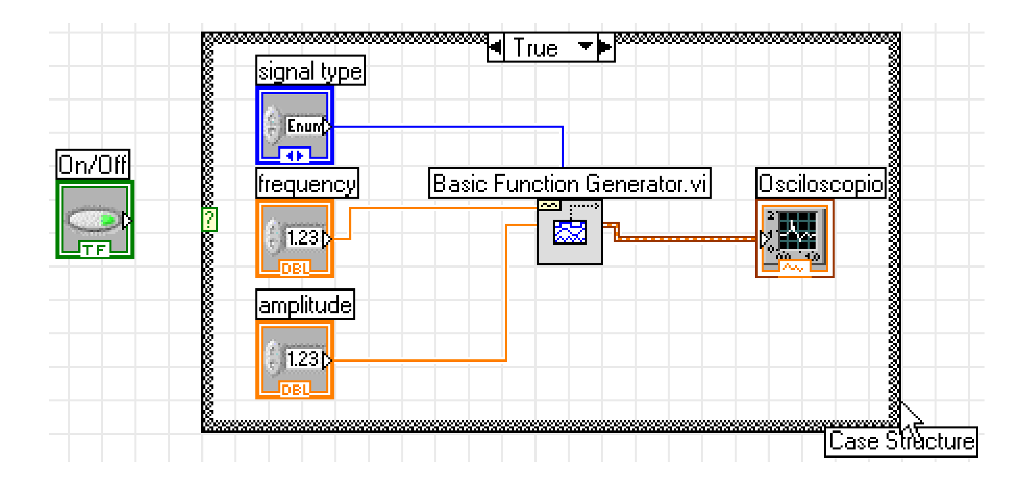
En el Diagrama de Bloques, conecte el terminal de salida *signal out* del **Basic FuncGen** al Osciloscopio. Ahora se colocará un botón de encendido y apagado para todo el circuito, incluye al Generador de Funciones y el Osciloscopio. En la Paleta de Controles seleccione y arrastre un botón del tipo **Push Button** desde *Modern→ Boolean* hasta el Panel Frontal.

Este control maneja datos de tipo Bolean: Verdadero/Falso, se emplea para controlar estructuras del tipo *Case*, entre otras. Para obtener la estructura tipo *Case* vaya al Diagrama de Bloque, en la Paleta de Funciones siga la secuencia *Programming* → *Structures* y seleccione **Case Structure.**

Cuando se encuentre sobre la pantalla del Diagrama de Bloque, el cursor tendrá la forma de un pequeño cuadrado con líneas segmentadas con la esquina superior izquierda rellena. Seleccione arrastrando todos los elementos del Diagrama de Bloque que desea estén controlados por la estructura *Case*.



Vista previa a la colocación de las estructura *Case*



Vista luego de colocar la estructura *Case.*

Conecte el botón On/Off a la estructura, cableando desde el triángulo del **Push Button** hasta el signo de interrogación que encuentra al lado izquierdo de la estructura. Note que el *Case* tiene un rectángulo en la parte superior, éste le permite ver el programa que se ha de ejecutar si la condición del botón es Verdadero (True) o Falso (False). La estructura también podría ser controlada por variables diferentes a la Boolean, si tiene un control de tipo numérico, entonces las condiciones del *Case* cambiarán automáticamente a las nuevas condiciones. Revise la Ayuda para obtener mayor información.

Así se verán las diferentes pantallas:

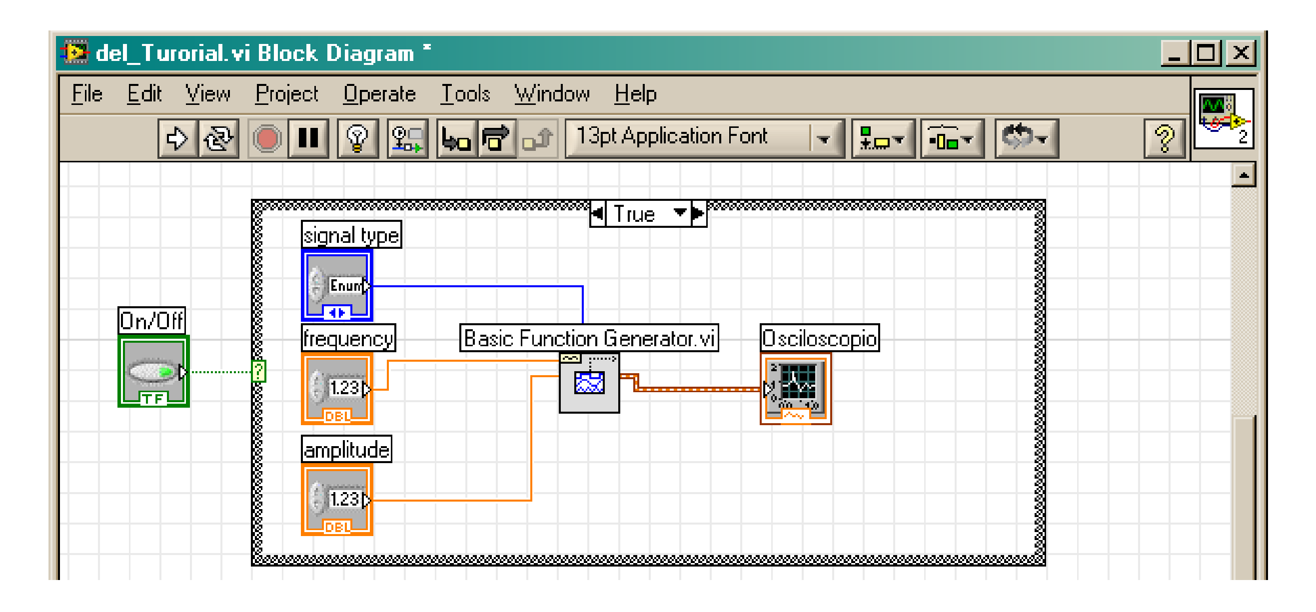
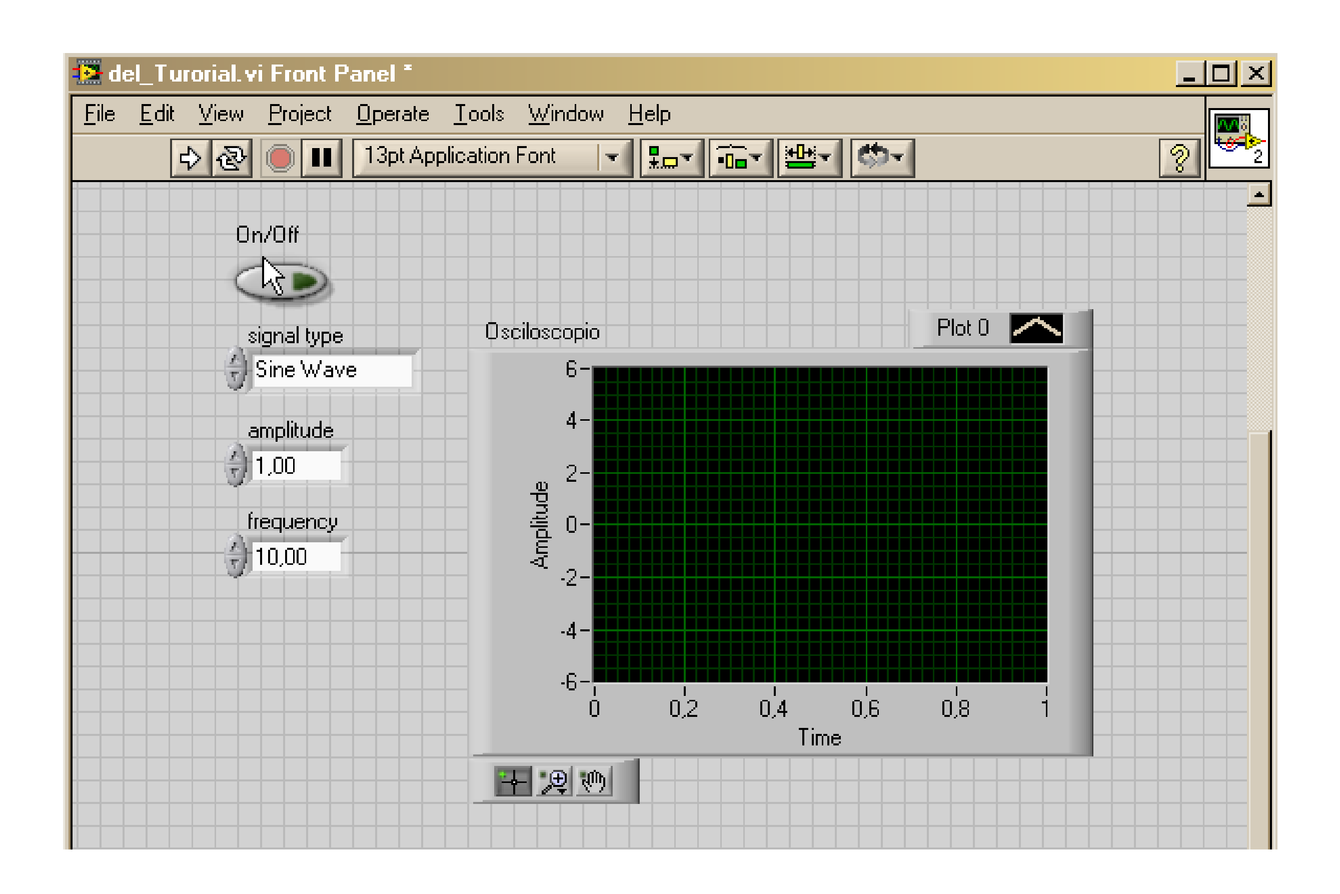


Diagrama de Bloques

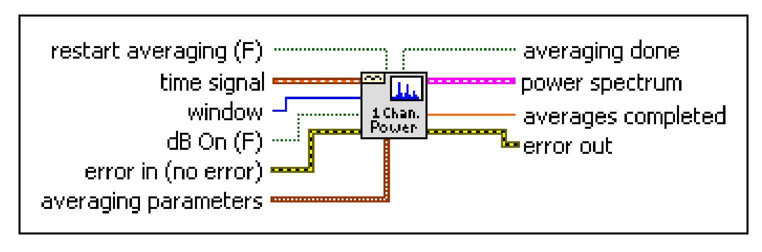


Panel Frontal

Desde el Panel Frontal presione el botón de ejecución continua, luego el botón On/Off. Ahora puede interactuar con el Generador de Funciones cambiando el tipo de Señal, la amplitud y/o la Frecuencia. Pruebe cambiar la escala, de amplitud y tiempo, del Osciloscopio.

El próximo paso será colocar el Analizador de Espectro, repita el procedimiento empleado para colocar un indicador gráfico (figura 37), pero esta vez asígnele el nombre de “Analizador de Espectro”. Asegúrese que el control en la pantalla del Diagrama de Bloques quede dentro de la estructura *Case.*

Desde la Paleta de Funciones localice el VI **FFT Power Spectrum.vi** a través de *Signal Processing → Waveform Measurements*. Selecciónelo y arrástrelo dentro de la estructura *Case*. Tome nota de las propiedades de este VI, especialmente de la unidad en que está expresada la salida *power spectrum.* Conecte el terminal de entrada *time signal* del **FFT Power Spectrum** al cable de conexión del Osciloscopio. Conecte, también, el terminal de salida *power spectrum* al control del Analizador de Espectro, note como cambia automáticamente de color el control del analizador.



FFT Power Spectrum

Vuelva a correr el programa y note la señal en tiempo y en frecuencia para cada tipo de función: Sinusoidal, Diente de Sierra, Señal de onda Cuadrada y Señal de onda Triangular. Para cada una de ellas varíe la frecuencia y la Amplitud.

Por último, investigue cómo reemplazar los controles de Amplitud y Frecuencia del Generador de Funciones por controles tipo Perilla (Sugerencia: en el Panel Frontal haga clic botón derecho sobre los controles que desea reemplazar y explore el menú que se despliega).

Al realizar los pasos anteriores debe obtenerse un producto como el que se presenta a continuación:

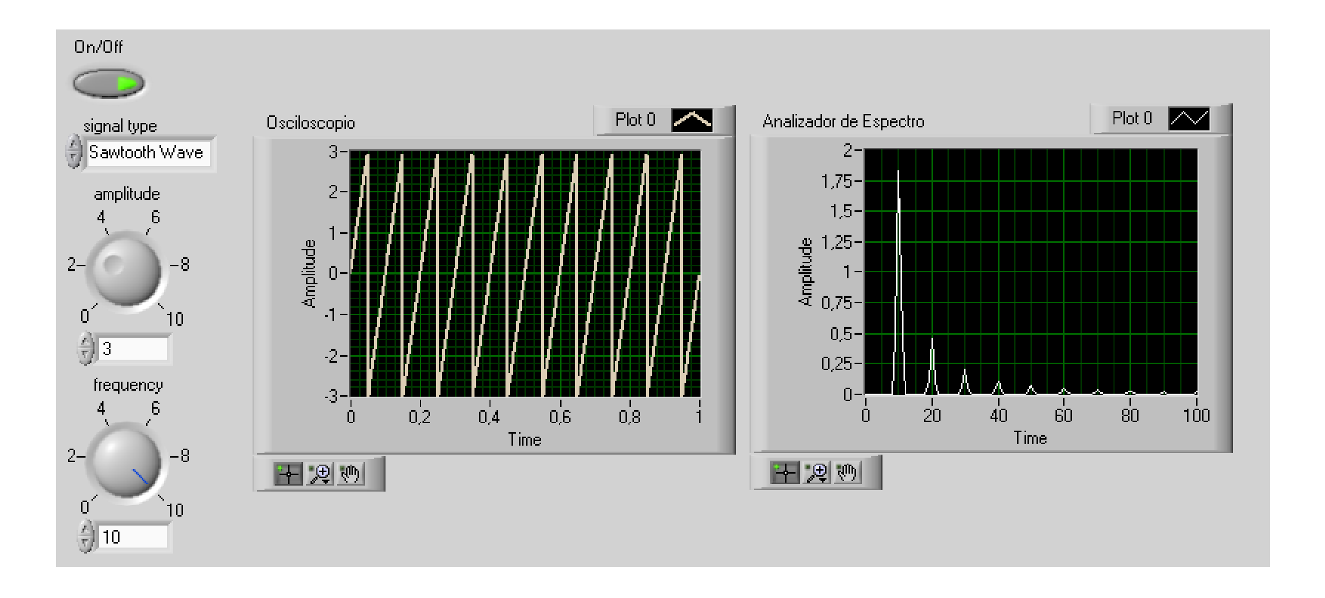


Imagen del Panel Frontal

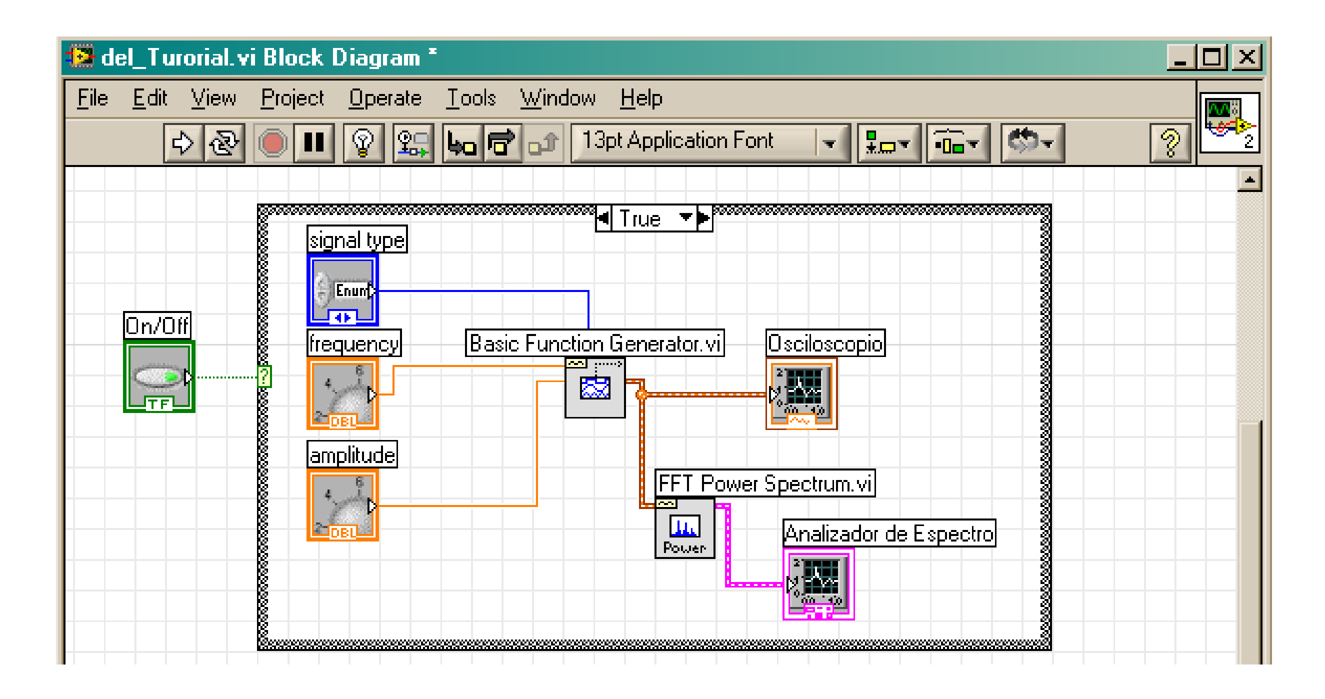


Imagen del Diagrama de Bloques

Introducción a MATLAB, comandos básicos y uso de GUIDE
Tutorial de MATLAB. Se incluye un tutorial acerca comandos y funciones básicas. Los tutoriales acerca de MATLAB y GUIDE en este módulo son los usados en la cátedra Señales y Sistemas Continuos de la UCAB, y fueron realizados por la profesora Trina Adrian, al igual que los archivos tutor1Trina.m y TutorialPractica1.m

MATLAB (MATrix LABoratory) es un entorno de programación basado en matrices, ampliamente utilizado en el ámbito de la ingeniería por su versatilidad en la resolución de cálculos numéricos sin importar cuán complejos sean. Se caracteriza también por permitir visualizar los resultados fácilmente y de distintas formas.

El uso de MATLAB está bastante extendido en la rama de Comunicaciones, debido a que permite el modelado de cualquier tipo de señales por medio de funciones, las cuales, al igual que todas las variables definidas, son representadas por matrices numéricas, que pueden incluir elementos complejos. MATLAB además incorpora ciertas librerías denominadas *toolboxes*, que no son más que conjuntos de funciones diseñados para la resolución de problemas específicos; en particular el *Communications Toolbox* y el *Signal Processing Toolbox* son muy útiles para simular la generación, procesamiento, transmisión y recepción de señales.

La ventana de comandos *(command window)*, ventana principal de MATLAB, guarda cierta semejanza con el Shell de un sistema operativo ya que, al igual que éste, permite ejecutar instrucciones, moverse entre directorios y gestionar el software en general. Para obtener ayuda sobre el uso de funciones predefinidas puede utilizarse la instrucción **help [comando]**. Siempre que una instrucción cree una variable, la misma se almacena en el *workspace*. Las variables se pueden guardar para futuras sesiones y se almacenarán en formato .mat, utilizando el comando **save**, y pueden ser cargadas de nuevo utilizando el comando **load**.

MATLAB permite el almacenamiento de secuencias de instrucciones en archivos por medio del editor de texto. Dichos archivos reciben el nombre de **archivos .m** debido a que se guardan con esta extensión. Al ejecutar un archivo .m, se ejecutarán una por una las instrucciones almacenadas por el

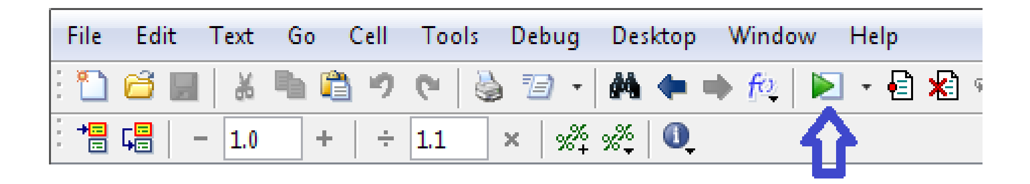
mismo, lo que permite que se creen programas y funciones capaces de cumplir tareas específicas.

Se incluye en ESTE VINCULO un documento que incluye una explicación rápida acerca de los **principales comandos y bloques utilizados en MATLAB**. ESTE VINCULO contiene un archivo para ejecutarse con MATLAB (de extensión .m) en el que se ponen en práctica algunos de los **comandos y operaciones básicas**. Por último, ESTE VINCULO contiene otro archivo .m en el que se pone en práctica la **generación de funciones y las distintas formas de graficarlas.**

Si se es principiante en MATLAB, es recomendable revisar previamente el documento y los archivos antes de aprender a trabajar con GUIDE.

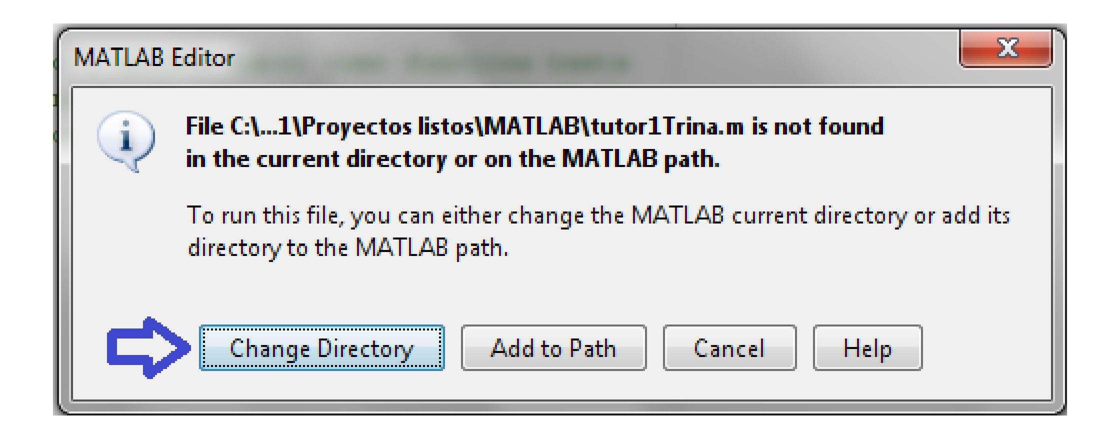
EJECUTAR PROGRAMAS EN MATLAB

El entorno de programación MATLAB cuenta con un editor en el que se construye el script de cada programa. En la parte superior del editor se pueden apreciar los menús de la figura 1; el botón indicado con la flecha azul sirve para ejecutar el programa.



Botón para correr el archivo de extensión .m

Cuando un programa se descarga, o simplemente es guardado en otro sitio que no es la carpeta por defecto de MATLAB, al instante de correrlo aparecerá un aviso como el mostrado en la figura 2, la opción más recomendada es la de Cambiar Directorio, señalada con la flecha azul:



Aviso para cambio de directorio

GUIDE

GUIDE es un ambiente de desarrollo que permite crear interfaces gráficas con el usuario, que contengan elementos tales como botones y ventanas de selección, ventanas gráficas, menús, ejes para graficar, etc.

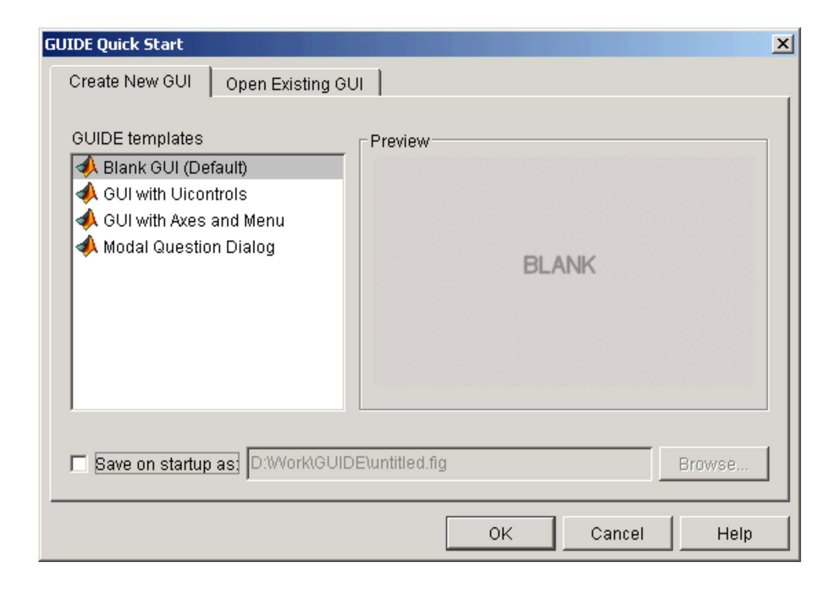
Cuando en el ´command window´ se escribe guide, se ofrece la posibilidad de abrir hojas de trabajo ya creadas (p.ej. >>guide archivo.fig) o una nueva sobre la cual se irán agregando componentes. Lo que se cree aquí se guardará con la extensión .fig.

La primera vez que uno salva la interfaz que está diseñando se crea también un archivo .m sobre el cual habrá que programar lo que se quiere ver o controlar desde el GUI

Una vez que se diseña la interfaz gráfica (GUI) que uno desea fijando las características de botones, ventanas, etc. que la conforman, se puede entonces programar dicha interfaz con el editor de archivos .m

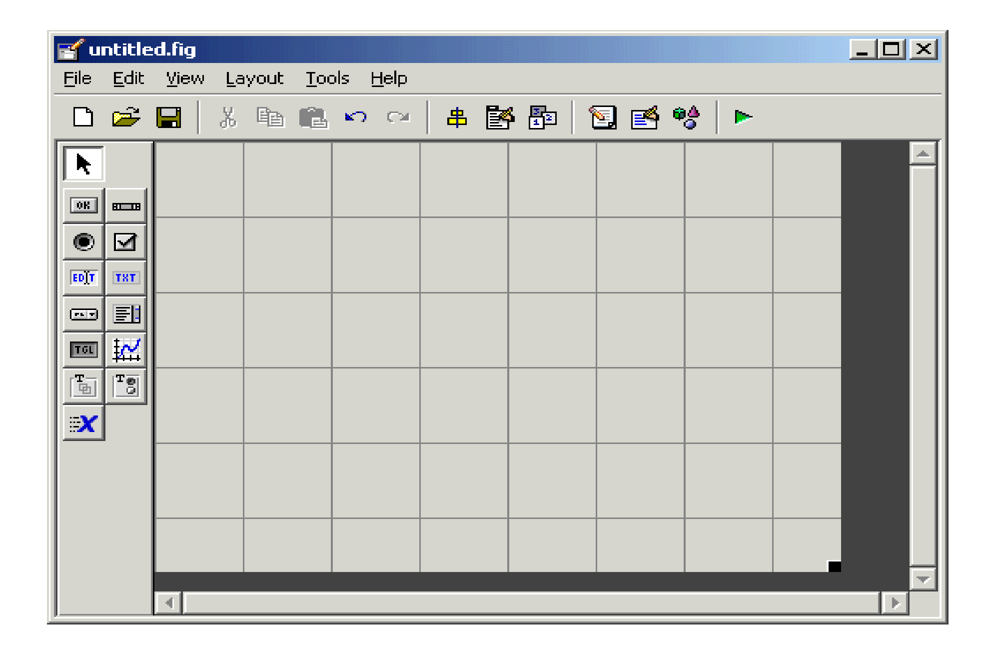
Herramientas:

Se selecciona una hoja en blanco (Blank GUI)



GUIDE Quick Start

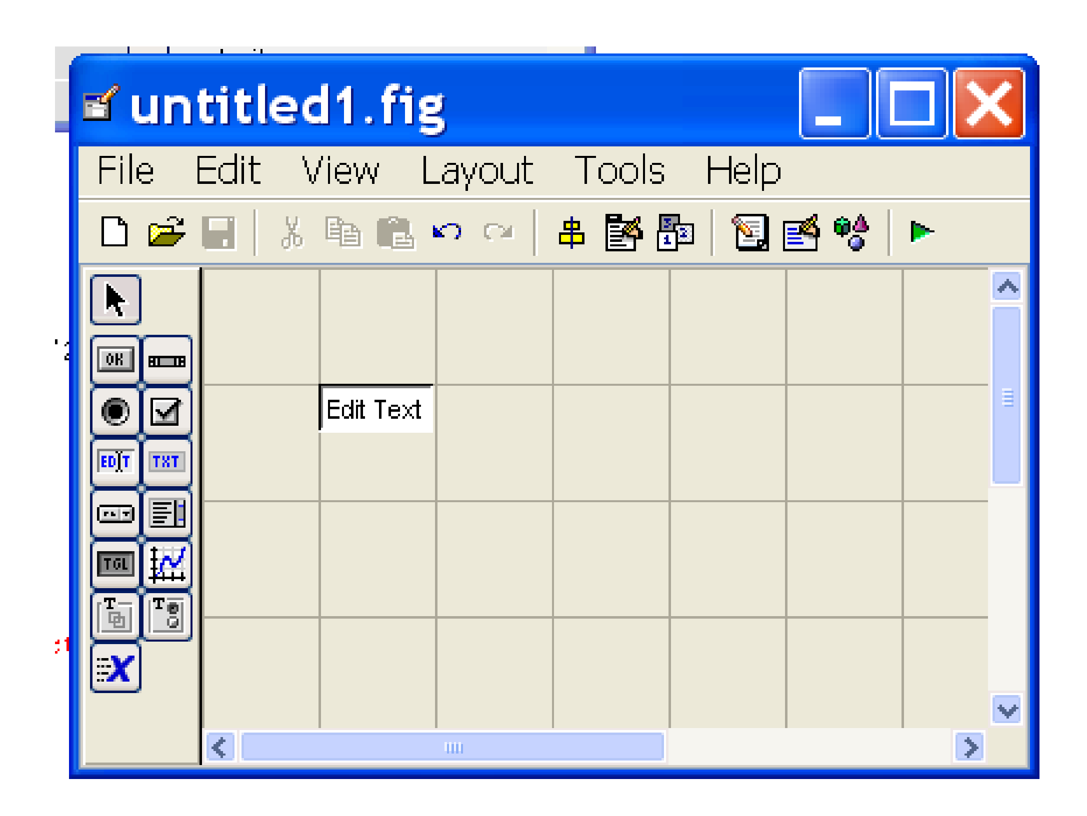
Aparecerá de inmediato una hoja de trabajo como la que sigue:



Espacio de trabajo aún sin elementos

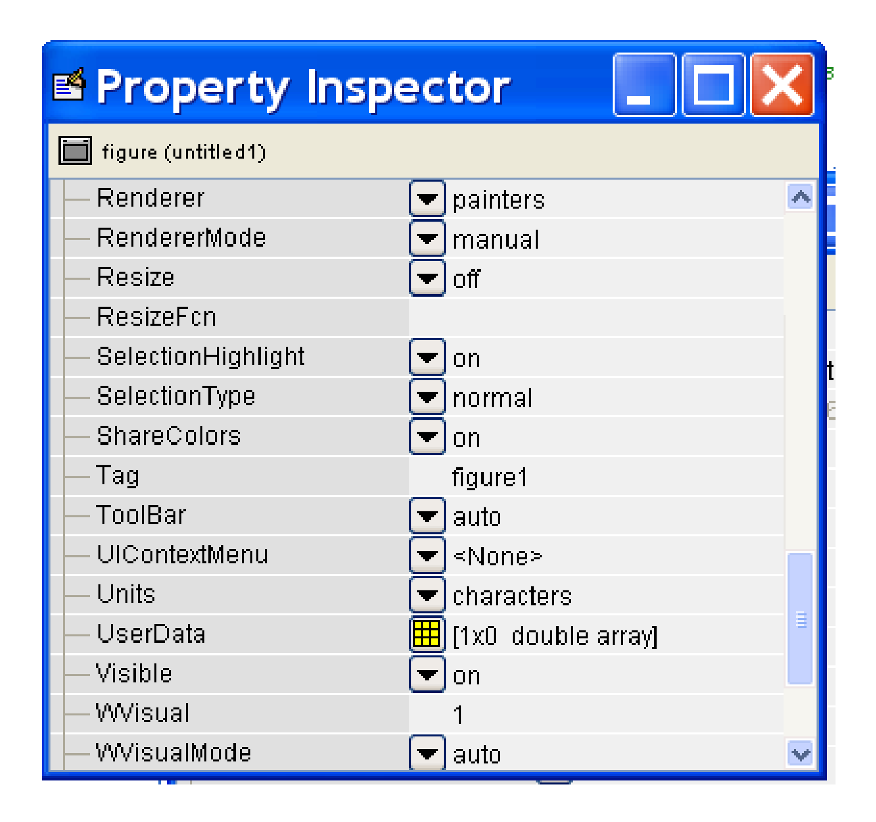
A la izquierda están las herramientas disponibles. Si en este momento se salva el GUI debe seleccionarse un nombre. Se almacenarán en el directorio que se seleccione los archivos nombre.fig y nombre.m

Para utilizar una herramienta se selecciona y arrastra a la posición deseada en la hoja de trabajo; se puede modificar su tamaño tal y como se hace con cualquier figura.



Espacio de trabajo con un solo elemento de tipo "Edit Text"

Otras características se pueden modificar haciendo doble click. Aparecerá un editor de propiedades como el que sigue:



Property Inspector

Al hacer una modificación en el editor de propiedades también cambiará el código relacionado a cada botón en el archivo .m

Una propiedad importante es el Tag (ver Property Inspector) el cual es el nombre que aparecerá en el Callback que no es más que una especie de “rutina” en el .m que se ejecutará cuando el elemento al que corresponde es manejado o cambiado su valor. Se recomienda que se coloque al tag un nombre correspondiente o alusivo a la acción del elemento.

El archivo nombre.m tiene toda una estructura de handles (manejadores) que alimentarán a la GUI. La estructura de handles es pasada como una entrada a cada callback (llamada a una parte de un programa).

Puede usarse la estructura de handles para:

- Compartir datos entre callbacks

- Acceder la data en el GUI

Para almacenar los datos contenidos en una variable X, se fija un campo de la estructura de handles igual a X y se salva la estructura de handles con guidata como se muestra a continuación:

handles.x=X
guidata(hObject, handles)

En cualquier momento se puede recuperar la data en cualquier callback con el comando:

X=handles.x

Código asociado a cada elemento del GUI

Automáticamente al crear el GUI y salvarlo aparece en el archivo .m una cantidad de líneas de código fijas.

En la primera parte del script aparece una cantidad de líneas de código fijo. La primera instrucción "function varargout = untitled(varargin)" indica que se está creando un GUI de nombre “untitled” con argumentos de salida “varargout” y argumentos de entrada “varagin”. Solo se muestran las dos primeras y la última línea.

```
function varargout = untitled(varargin)
% UNTITLED M-file for untitled.fig
........
% Last Modified by GUIDE v2.5 28-Feb-2004 08:46:04
```

Código generado al crear y guardar el GUI

Aquí comienza un código de inicialización que se pide no se edite. También se muestra solo la primera y última línea.

```
% Begin initialization code - DO NOT EDIT

.........

% End initialization code - DO NOT EDIT
```

Código fijo de inicialización

Hasta aquí llega el código de inicialización. Comienza entonces lo que se desea que ocurra antes de que el GUI se haga visible.

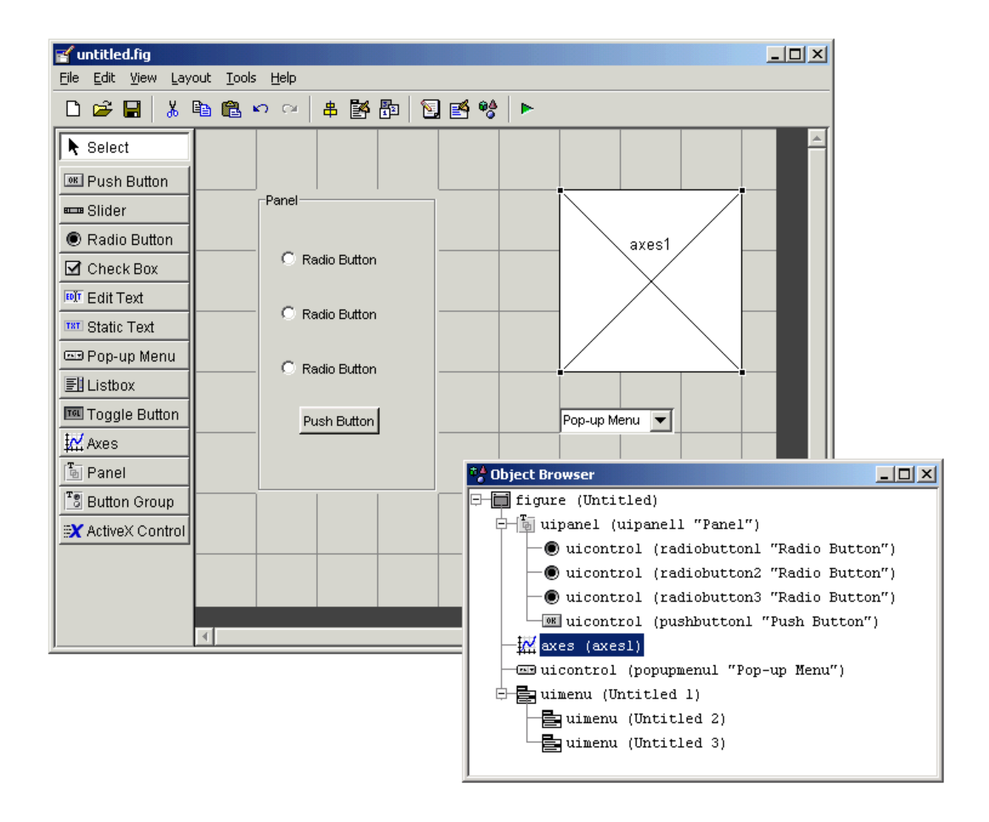
```
% --- Executes just before untitled is made visible.
function untitled_OpeningFcn(hObject, eventdata, handles, varargin)

        %% COLOCAR AQUÍ EL CÓDIGO QUE SE ESPERA EJECUTAR ANTES DE
        %% QUE EL GUI SE HAGA VISIBLE

varargout{1} = handles.output;
```

Función OpeningFCN para un programa de nombre "untitled"

Luego de esto aparecen callback dependiendo de las herramientas que se han incorporado al GUI. Por ejemplo:



Espacio de trabajo ya con varios elementos

Aquí se han colocado 3 Radio Button, 1 Push Button, un Pop-Up menú y 1 eje para graficar.

A continuación se describen brevemente las herramientas disponibles, el código automático asociado a cada una de ellas y como interactuar con las mismas.

Botón pushbutton

Se ejecuta una determinada acción cuando son presionados. En el archivo .m aparecen automáticamente un grupo de instrucciones asociadas a él.

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
        %% AQUI SE COLOCA EL CODIGO QUE SE ESPERA EJECUTAR CUANDO
        %% SE PRESIONE EL PUSHBUTTON1
% hObject    handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Callback para elemento de tag "pushbutton1"

Listas de selección

Aquí se puede colocar una lista de elementos para que el usuario pueda seleccionar alguno. En el archivo .m aparecen automáticamente un grupo de instrucciones asociadas a él.

```
% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)

.........
```

Parte del código generado por elemento de tag "listbox1"

```
% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)

..............
% Hints: contents = get(hObject,'String') returns listbox1 contents as cell array
% contents{get(hObject,'Value')} returns selected item from listbox1
```

Callback para elemento de tag "listbox1"

Como se observa en la ayuda (Hint) que aparece en las dos últimas líneas, si dentro del callback de este elemento se coloca la instrucción “A=get(hObject,'String')”, se obtendrá un número que indica que selección se hizo. Por ejemplo si se seleccionó el tercer elemento de la lista, A valdrá 3. Lo mismo puede lograrse con la instrucción “A=get(handles.listbox1,'Value')”, donde “listbox1” es el tag de este elemento. La ventaja de esta instrucción es que no tiene obligatoriamente que ser llamada desde el callback del elemento.

Botón edit

Permite a los usuarios ingresar o modificar parámetros que se quieren introducir.

```
function edit1_Callback(hObject, eventdata, handles)

.....
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
```

Callback para elemento de tag "edit1"

Obsérvese la ayuda (HINT). Si dentro del callback de este elemento se coloca la instrucción “A=str2double(get(hObject,'String'))”, entonces, se podrá tener en A el valor del número que se escribió en la casilla

Si dentro del callback se coloca la instrucción “A= get(hObject,'String')”, se podrá almacenar en A los caracteres escritos

Una forma más recomendable de almacenar en una variable A el contenido de la casilla es con la instrucción “A= get(handles.edit1,'String')”, donde “edit1” es el tag del botón. Esta instrucción puede ser colocada en otro callback que no sea el del botón o incluso en la función OpeningFCN de la figura 9. Si se desea acceder al valor numérico escrito en la casilla, se hace uso de la instrucción “A=str2double(get(handles.edit1,'String'))”.

Si se desea colocar el valor de la variable A en la casilla, se hace uso de la instrucción “set(handles.edit1,'String',A)”

Botón RadioButton

Son botones de selección. Si hay varios generalmente son mutuamente excluyentes. Para seleccionarlo basta ubicarse y presionar el ratón.

```
% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)

........
% Hint: get(hObject,'Value') returns toggle state of radiobutton1
```

Callback para elemento de tag "radiobutton1"

Colocándose dentro de callback del elemento la instrucción “A=get(hObject,'Value')”, A valdrá 1 si el botón fue seleccionado, y 0 en caso contrario. Lo mismo puede lograrse con la instrucción

“A=get(handles.radiobutton1,'Value')”, donde “radiobutton1” es el tag de este elemento. La ventaja de esta instrucción es que no tiene obligatoriamente que ser llamada desde el callback del elemento.

Ejes para graficar

Aquí no se genera nada especial en el archivo .m, pero uno debe fijar las condiciones de la gráfica y activarla o desactivarla según convenga.

Por ejemplo si la gráfica tiene asociado el nombre de axes1 (tag) y se quiere mostrar algo a través de la misma por medio de la instrucción "plot(...)" se debe agregar primero el comando "axes" como se muestra a continuación:

```
axes(handles.axes1);
plot(...)
```

PopUp menú

Cuando se hace click despliega opciones. Para agregar elementos a la lista, en el editor de propiedades se busca el elemento string y allí se coloca la lista de las opciones.

```
% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)

..........
```

Parte del código generado por elemento de tag "popupmenu1"

```
% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)

........
% Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1
```

Callback para elemento de tag "popupmenu1"

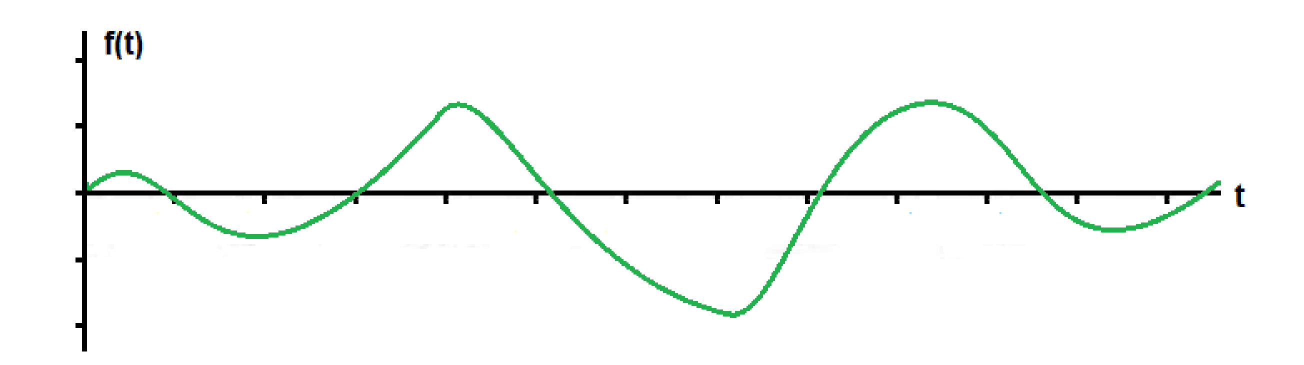
Si se coloca dentro del callback de este elemento la instrucción “A=get(hObject,'Value')”, se obtendrá un número que indica que selección se hizo. Por ejemplo si se seleccionó el tercer elemento de la lista, A valdrá 3. Lo mismo puede lograrse con la instrucción “A=get(handles.popupmenu1,'Value')”, donde “popupmenu1” es el tag de este elemento. La ventaja de esta instrucción es que no tiene obligatoriamente que ser llamada desde el callback del elemento.

Las Señales y sus diferentes clasificaciones
Se define el concepto de señal y se muestran sus diferentes clasificaciones. Incluye un programa en MATLAB y otro en LabVIEW, los cuales generan señales de diferente tipo y realizan operaciones entre ellas

Una señal puede definirse como la manifestación eléctrica de algún fenómeno natural, la cual toma valores de voltaje que varían en función del tiempo según el comportamiento de dicho fenómeno. Existen varias formas de clasificar las señales entre las que se encuentran: continuas y discretas, de energía y de potencia, periódicas y aperiódicas o determinísticas y aleatorias.

Señales Continuas y Señales Discretas

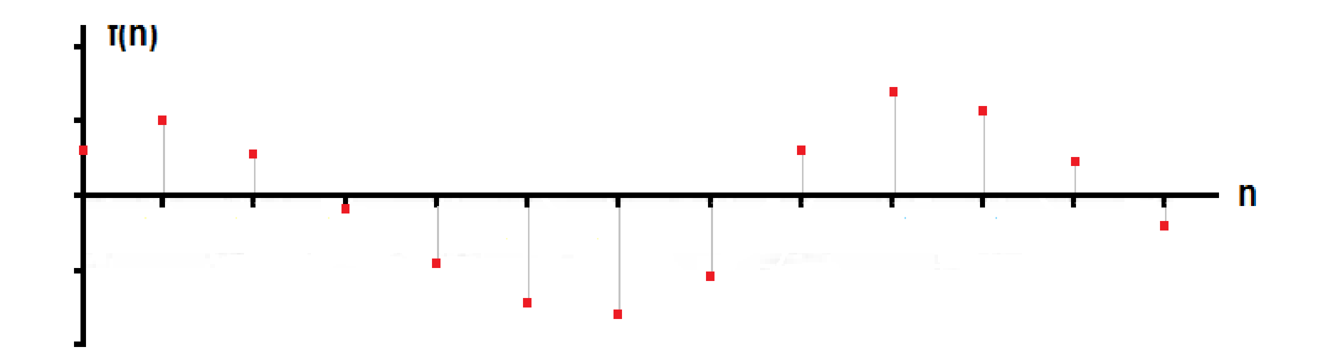
El dominio para para las señales continuas puede contener cualquier valor perteneciente a los números reales, y para cada uno de estos valores, la señal puede tomar cualquier valor real, normalmente comprendido entre un máximo y un mínimo; un ejemplo de señal continua se muestra en la figura 1:



Señal Continua

Las señales discretas pueden tomar cualquier valor real pero sólo existen para una cantidad limitada de valores los cuales normalmente se encuentran equiespaciados; una señal discreta puede venir de un proceso en el cual la

variable independiente de por sí es discreta, por ejemplo, el valor de la temperatura de cierto objeto medida cada minuto; o puede proceder del muestreo de una señal analógica o continua. La figura 2 muestra un ejemplo de señal discreta:



Señal Discreta

Señales de Energía y Señales de Potencia.

La energía y la potencia de una señal f(t) para un intervalo definido entre T1 y T2 vienen dadas por las ecuaciones 1 y 2 respectivamente:

Equation:

$$E_{T1 \to T2} = \int_{T1}^{T2} \mid f(t) \mid^2 \mathrm{dt}$$

Equation:

$$P_{T1 \to T2} = \frac{1}{T2 - T1} \int_{T1}^{T2} \mid f(t) \mid^2 \mathrm{dt}$$

Comúnmente será necesario cuantificar la energía y la potencia para un intervalo que de tiempo infinito, es decir, definido entre -infinito y +infinito. Para ello se definen las ecuaciones 3 y 4 por medio de límites quedando de la siguiente forma:

Equation:

$$E = \lim_{T \to \infty} \int_{-T}^{T} \mid f(t) \mid^2 \mathrm{dt}$$

Equation:

$$P = \lim_{T \to \infty} \left[\frac{1}{2\mathrm{T}} \int_{-T}^{T} \mid f(t) \mid^2 \mathrm{dt} \right]$$

Si f(t) se trata de una función existente para todo valor de t, como la señal periódica f(t) = sen(t), la integral de la ecuación 3 puede separarse en la suma de infinitas integrales en intervalos definidos similares a los de la ecuación 1, las cuales arrojarán resultados positivos, por lo que el valor de la energía total será infinito; si se realiza el mismo proceso de separación en la ecuación 4 se obtendrá la misma suma de una cantidad infinita de valores positivos pero dividida entre esa misma cantidad de valores (lo cual corresponde con el proceso realizado para el cálculo de promedios), resultando un valor de potencia finito y positivo denominado Potencia Promedio Total. Toda señal con una energía total infinita y con una potencia promedio total finita recibe la denominación de Señal de Potencia; en general, las señales de potencia serán aquellas no limitadas en tiempo.

Si f(t) se trata de una función existente sólo para un intervalo de valores de t (o para una limitada cantidad de intervalos), la integral de la ecuación 3 será nula para todo valor fuera del intervalo de existencia de f(t), por lo cual la energía total será un valor finito y positivo; para calcular la potencia promedio total se divide la cantidad obtenida entre infinito como lo indica la ecuación 4, dando como resultado un valor nulo. Toda señal con una potencia promedio total igual a cero y con una energía total finita recibe la

denominación de Señal de Energía; en general, pueden clasificarse en este grupo las señales limitadas en tiempo como por ejemplo un pulso rectangular que tiene un valor de 1 para$0 \leq t \leq 1$ y de 0 para cualquier otro caso.

Puede encontrarse otro tipo de señales para las cuales la energía y potencia promedio total son infinitas, por ejemplo la señal $f(t) = e^t$; en general se incluyen en este grupo aquellas señales que cumplan lo siguiente:
Equation:

$$\mid \lim_{t \to \infty} f(t) \mid = \infty$$

Todo lo anterior se cumple también para señales discretas; las ecuaciones para la Energía Total y para la Potencia Promedio Total se expresan en las ecuaciones 6 y 7 respectivamente:
Equation:

$$E = \sum_{n=-\infty}^{\infty} \mid f(n) \mid^2$$

Equation:

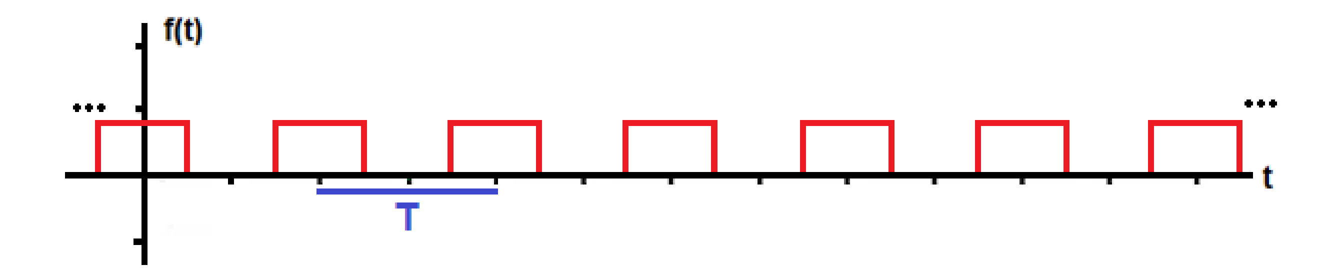
$$P = \lim_{N \to \infty} \left[\frac{1}{2\mathrm{N} + 1} \sum_{n=-N}^{N} \mid f(n) \mid^2 \right]$$

Señales Periódicas y Señales Aperiódicas

Las señales periódicas se encuentran entre las Señales de Potencia explicadas anteriormente y son aquellas que para todo valor de t cumplen con la igualdad en la ecuación 8:
Equation:

$$f(t) = f(t + \mathrm{kT})$$

Esto quiere decir que la señal no cambiará para un desplazamiento de tiempo T para todo valor entero de k positivo o negativo; dicho valor T se conoce como Período. En la figura 3 se muestra un ejemplo de señal periódica:



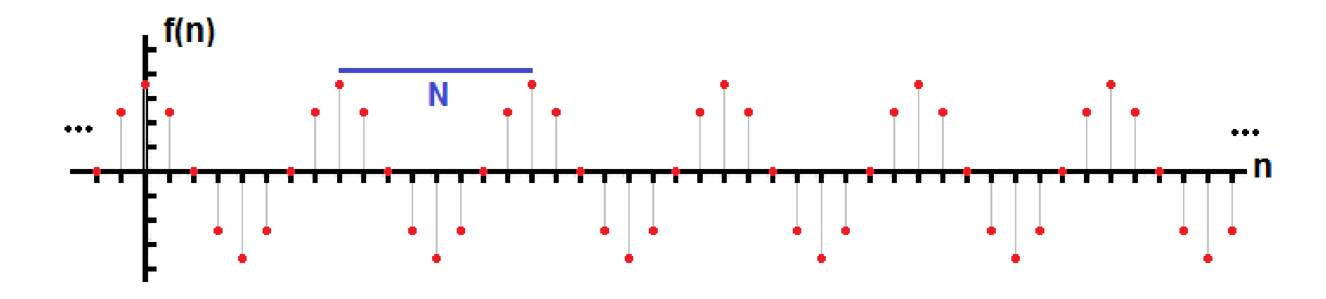
Señal Periódica Continua

De forma análoga, una señal discreta es periódica si cumple con la ecuación 9:

Equation:

$$f(n) = f(n + \mathrm{kN})$$

Donde N es un valor entero positivo correspondiente con el periodo de la señal, y k representa un valor entero que representa que la señal es periódica para cualquier múltiplo de N. La figura 4 muestra un ejemplo de señal periódica discreta con un período de N=8:



Señal Periódica Discreta

Las señales aperiódicas son simplemente aquellas que no son periódicas, es decir, no cumplen con las ecuaciones 8 o 9. Según la definición, una señal periódica tendría que estar definida en un intervalo de tiempo que va desde –infinito hasta infinito, esta es una situación ideal, de hecho, en la vida real se considera que una señal es periódica si su duración tiende a infinito con respecto a su período, por ejemplo, la señal en una línea eléctrica es una onda senoidal con un período de [1/60] segundos la cual sufre de cortes muy eventualmente, por lo que esta señal estará definida desde el momento en el que se recupera de un corte hasta el momento en el que ocurre otro corte, un tiempo que tiende a infinito comparado con el período; las señales también pueden ser periódicas para un tiempo limitado, pero que sea el tiempo total de duración de cierto evento, por ejemplo, una señal cuadrada similar a la de la figura 4 usada para mantener la sincronización en un dispositivo electrónico sólo está definida cuando dicho dispositivo esté encendido y es nula el resto del tiempo, se considera periódica a esta señal ya que el tiempo en el que el dispositivo no esté encendido no entra en el análisis.

Señales Determinísticas y Señales Aleatorias

Las señales determinísticas pueden ser modeladas por medio de una expresión matemática totalmente determinada, mientras que las aleatorias pueden ser modeladas por medio de la *Función de Densidad de Probabilidades*, o en el mejor de los casos, por medio de la función de *Autocorrelación.* Como se muestra en la ecuación 10, podría construirse una expresión matemática para alguna señal aleatoria, la misma puede involucrar una variable independiente (el tiempo, por ejemplo) y una variable aleatoria de la que podría conocerse la función de densidad de probabilidades.

Equation:

$$x(t) = \cos(2\pi t + \theta)$$

Considerándose que esta función está definida para $t \geq 0$, θ será la variable aleatoria que representa el valor de fase que puede tener la señal para $t=0$; esto se interpreta como el hecho de que al momento de encender un generador de funciones, el valor de la fase puede ser cualquiera entre 0 y 2π. *Véase* procesos aleatorios y sus elementos.

Autoevaluación

Exercise:

Problem:

¿Todas las señales de energía están acotadas o limitadas en tiempo?

Solution:

No. Una excepción sería por ejemplo Sinc(t). Esta señal es de Energía pero ilimitada en tiempo. Si se observa en el **dominio de la frecuencia** se entenderá que las señales acotadas o limitadas en frecuencia también son de energía.

Exercise:

Problem:

Si se suma una señal de potencia más una señal de energía, ¿Resultará una señal de potencia o de energía?

Solution:

Al sumar dos señales (una de potencia y la otra de energía), la energía de esta nueva señal será infinita, por lo tanto esta nueva señal NO es de energía.

Exercise:

Problem:

¿La señal aleatoria conocida como **ruido blanco** es una señal de potencia o energía?

Solution:

Su energía es ilimitada, en cambio la potencia si tiene un valor finito. Esto indica que es una señal de potencia.

Exercise:

Problem:

Si se suma una señal periódica $x_1(t)$ con T=4 con otra señal periódica $x_2(t)$ con T=6, ¿cuál será el período (T) de la señal resultante?

Solution:

Supóngase que el análisis se comienza en t=0. Si se determina el mínimo común múltiplo de los valores de T de las señales a sumar, se obtendrá el valor de t donde ambas señales comenzarán un nuevo ciclo como ocurre en t=0, por lo cual este será el valor del período de la señal resultante; el mínimo común múltiplo entre 4 y 6 es 12. Obsérvese que en doce segundos han transcurrido dos periodos de x_2 y tres periodos de x_1.

Simuladores

ESTE VINCULO contiene una carpeta con un programa realizado en **MATLAB** capaz de generar señales de diversos tipos, además de aplicar operaciones entre ellas, como suma, multiplicación o **convolución**. La carpeta incluye el .m y todos los archivos necesarios para su funcionamiento, si se elimina o renombra alguno de estos archivos, el programa podría no funcionar correctamente. La figura 5 contiene un video explicativo acerca del uso del programa.

Generador MATLAB

[missing_resource: http://www.youtube.com/v/ydqIspRxBM4?fs=1&hl=es_ES]

Video explicativo de la utilización del programa realizado en MATLAB

Puede obtenerse también un programa realizado en **LabVIEW** acerca del mismo tema por medio de ESTE VINCULO. La carpeta incluye el .vi y todos los archivos necesarios para su funcionamiento. Igualmente, si se elimina o renombra alguno de estos archivos, el programa podría no funcionar correctamente. La figura 6 contiene un video explicativo acerca del uso del programa.

Generador LabVIEW

[missing_resource: http://www.youtube.com/v/IPIP1C5Me9g?fs=1&hl=es_ES]

Video explicativo de la utilización del programa realizado en LabVIEW

Los Sistemas y sus diferentes clasificaciones
Se define el concepto de sistema y se muestran sus diferentes clasificaciones

Un sistema hace referencia a cualquier medio físico que modifique las características de una **señal**; el mismo puede tratarse de algún dispositivo electrónico como un filtro, un amplificador, entre otros, como también puede tratarse del canal por el que se transmite la señal como el aire o los cables. Los sistemas se modelan por medio de la **Respuesta Impulsiva h(t)**, una función en el dominio del tiempo que representa la salida del sistema cuando la entrada es la función delta de Dirac; y por medio de la **Función de Transferencia H(ω)**, una función en el dominio de la frecuencia que corresponde con la **Transformada de Fourier** de la respuesta impulsiva. Si una señal es procesada por un sistema, puede **convolucionarse** la respuesta impulsiva del sistema con la expresión temporal de dicha señal y se obtendrá la expresión temporal de la señal de salida; puede obtenerse también la expresión frecuencial de la señal de salida multiplicando la respuesta impulsiva del sistema por la expresión frecuencial de la señal de entrada.

Equation:

$$y(t) = h(t) * x(t)$$

Equation:

$$Y(\omega) = H(\omega)X(\omega)$$

Gráficamente, los sistemas suelen representarse de la siguiente forma:



Representación gráfica de un sistema

Existen varias formas de clasificar los sistemas, entre las que se encuentran las siguientes:

Sistemas Lineales y No Lineales

Si la salida de un sistema es igual a $y_1(t)$ cuando se estimula con una la señal $x_1(t)$ y es igual a $y_2(t)$ cuando se estimula con una la señal $x_2(t)$, dicho sistema será lineal si se cumple que cuando se estimula con la señal $x_1(t) + x_2(t)$, su salida será igual a $y_1(t) + y_2(t)$; esto se conoce como principio de superposición. Para la linealidad se debe cumplir también que si la señal de entrada se multiplica por una constante, la salida se multiplicará también por dicha constante.

Como primer ejemplo, supóngase un sistema que duplica la amplitud de la señal entrante:

Equation:

$$\mathrm{x}_1(t) \rightarrow y_1(t) = 2\mathrm{x}_1(t)$$
$$3\mathrm{x}_2(t) \rightarrow y_2(t) = 6\mathrm{x}_2(t)$$

Para que el sistema sea lineal debe cumplirse que su salida sea igual a:

Equation:

$$y_1(t) + y_2(t) = 2\mathrm{x}_1(t) + 6\mathrm{x}_2(t)$$

Si el sistema se alimenta con $x_1(t) + 3x_2(t)$, en la salida se obtiene:

Equation:

$$\mathrm{x}_1(t) + 3\mathrm{x}_2(t) \rightarrow 2(\mathrm{x}_1(t) + 3\mathrm{x}_2(t)) = 2\mathrm{x}_1(t) + 6\mathrm{x}_2(t)$$

Como puede observarse, las ecuaciones 4 y 5 arrojan el mismo resultado, de donde se concluye que el sistema es lineal. Como un segundo ejemplo, supóngase un sistema que eleva al cuadrado la señal de entrada:

Equation:

$$x_1(t) \to y_1(t) = (x_1(t))^2$$
$$x_2(t) \to y_2(t) = (x_2(t))^2$$

Para que el sistema sea lineal debe cumplirse que su salida sea igual a:
Equation:

$$y_1(t) + y_2(t) = (x_1(t))^2 + (x_2(t))^2$$

Si el sistema se alimenta con $x_1(t) + x_2(t)$, en la salida se obtiene:
Equation:

$$x_1(t) + x_2(t) \to (x_1(t) + x_2(t))^2$$

Al no ser iguales las ecuaciones 7 y 8 el sistema no es lineal.

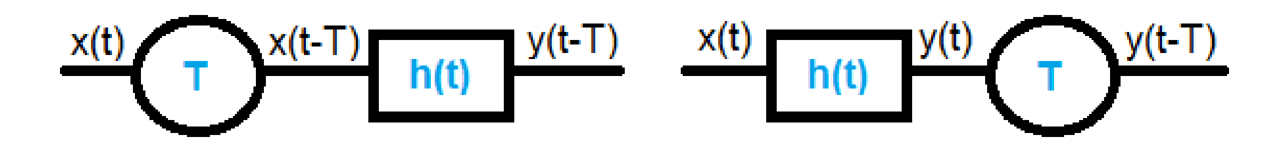
Sistemas Causales y No Causales

Los sistemas causales son también conocidos como sistemas No anticipativos, esto quiere decir que su salida no depende del futuro de la señal de entrada, sólo depende del presente o del pasado. Por ejemplo, un sistema que al ser estimulado con una señal x(t), su salida y(t) es igual a x(t+2) será un sistema no causal, ya que para t=0 su entrada es x(0) y su salida es x(2), es decir la salida depende del futuro de la señal de entrada. Si por el contrario se tiene un sistema que al ser estimulado con una señal x(t), su salida es igual a x(t-2), dicho sistema será causal, ya que en este caso la salida depende del pasado de la señal; un sistema que duplica la amplitud de la señal de entrada (x(t)→2x(t)) es un sistema causal, ya que la salida depende en este caso del presente de la señal de entrada.

Sistemas Variantes e Invariantes en el Tiempo.

Un sistema es invariante en el tiempo si se cumple que: x(t-T)→y(t-T), esto quiere decir que si hay un retardo T, será igual si el mismo se aplica a la

señal antes de pasar por el sistema o si se aplica luego de pasada por el sistema, como se ilustra en la Figura 2:



Sistema Invariante en el Tiempo

Si no se cumple esta propiedad, se dice que el sistema es Variante en el Tiempo; como ejemplo, supóngase un sistema con un comportamiento como el siguiente:

Equation:

$$x(t) \rightarrow y(t) = t \cdot x(t)$$

Si la señal se retarda primero y luego se pasa por el sistema se obtiene:

Equation:

$$t \cdot x(t - T)$$

Si la señal se pasa por el sistema primero y luego se retarda se obtiene:

Equation:

$$(t - T) \cdot x(t - T)$$

Como puede observarse, el resultado para ambos casos difiere, de donde se concluye que el sistema es variante en el tiempo.

Sistemas Estables e Inestables.

Un sistema estable es aquel cuya salida no diverge cuando es alimentado con una señal acotada; esto es resumido en la ecuación 12:
Equation:

$$| x(t) | \leq M \rightarrow y(t) < \infty$$

Un ejemplo de sistema inestable es el descrito en la ecuación 9, ya que aun si x(t) es una señal acotada (como cos(t), por ejemplo), la salida divergirá para $t \rightarrow \infty$. Si en cambio, el sistema se trata de un duplicador de amplitud, el mismo es estable, ya que si se alimenta con una señal acotada como cos(t), la salida tendrá valores entre -2 y 2 para todo valor de t.

Convolución
La convolución y sus propiedades

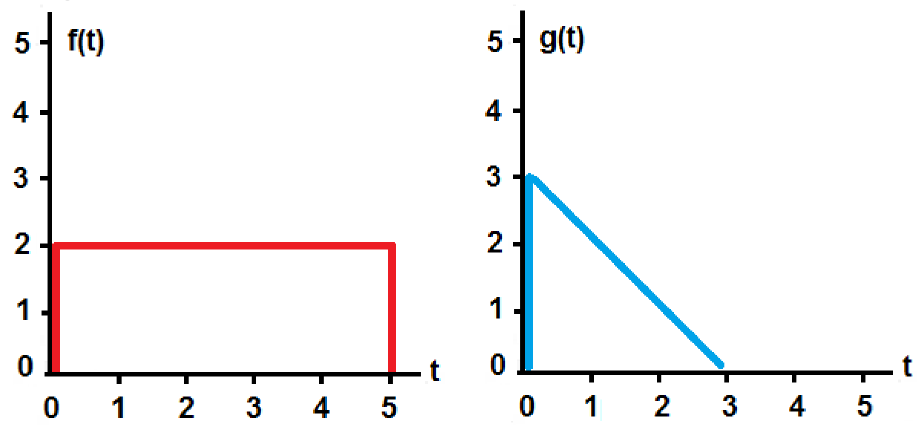
Es una herramienta temporal para la resolución del problema del paso de **señales** por **sistemas** que aplica para el caso específico en el que el sistema sea Lineal e Invariante en el Tiempo. Si se tiene la señal en el dominio del tiempo y la respuesta impulsiva del sistema, se puede aplicar la operación convolución entre ambos y se obtiene la salida del sistema en el dominio del tiempo.

Se define la convolución entre dos funciones f(t) y g(t) como el área bajo la curva formada por el producto de las mismas luego de invertir una de ellas y desplazarla una cantidad de tiempo que varía entre –infinito e infinito; la expresión para la convolución viene dada por la ecuación 1:

Equation:

$$y(t) = f(t) * g(t) = \int_{-\infty}^{\infty} f(\tau) g(t-\tau) d\tau$$

Una mejor forma de entender este proceso es haciendo el análisis gráfico del mismo, para ello supónganse las dos funciones f(t) y g(t) mostradas en la figura 1:



Funciones a convolucionar

Las ecuaciones para cada uno de los pulsos de la figura 1 son:

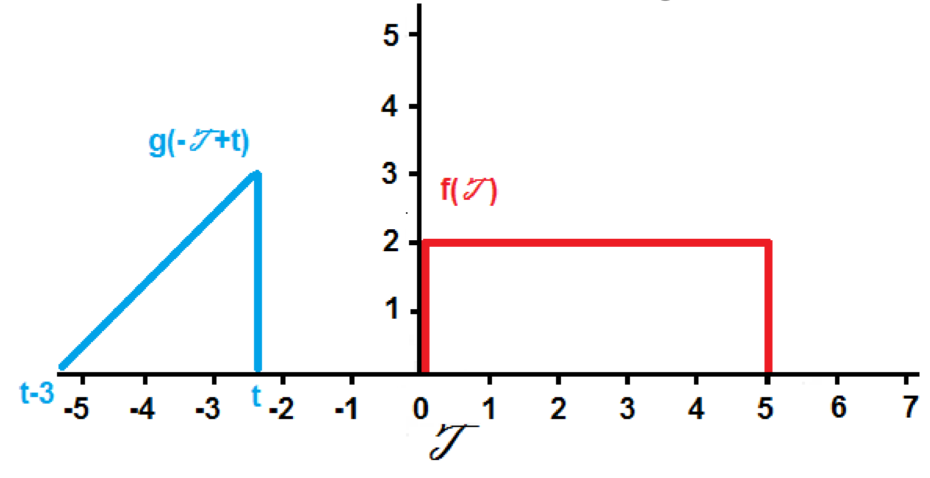
Equation:

$$f(t)=2\Rightarrow 0\leq t\leq 5$$

Equation:

$$g(t)=-t+3\Rightarrow 0\leq t\leq 3$$

Para comenzar se debe crear una función similar a f(t) pero expresada en términos de la variable τ presente en la ecuación , también se debe crear una función similar a g(t) pero con la variable τ negativa, además desplazada una cantidad t, tal como se indica en la figura 2:



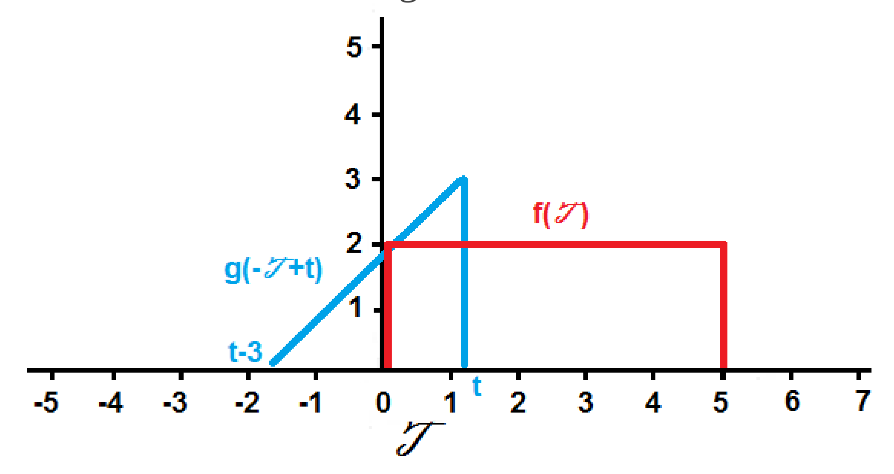
funciones similares en términos de τ

La cantidad t irá tomando valores desde –infinito hasta infinito, lo que causará que la función g haga un recorrido completo por el eje τ, en dicho recorrido se multiplican ambas funciones y se toma el área bajo el producto, por lo que la convolución valdrá 0 en todos los puntos donde las funciones no se intersecten, como sucede en la figura 2. Con esto ya se puede concluir que, para este caso:

Equation:

$$y(t)=0\Rightarrow t<0$$

Desde el instante en el que t es igual a 0 hasta que es igual a 3 se dará la situación descrita en la figura 3:



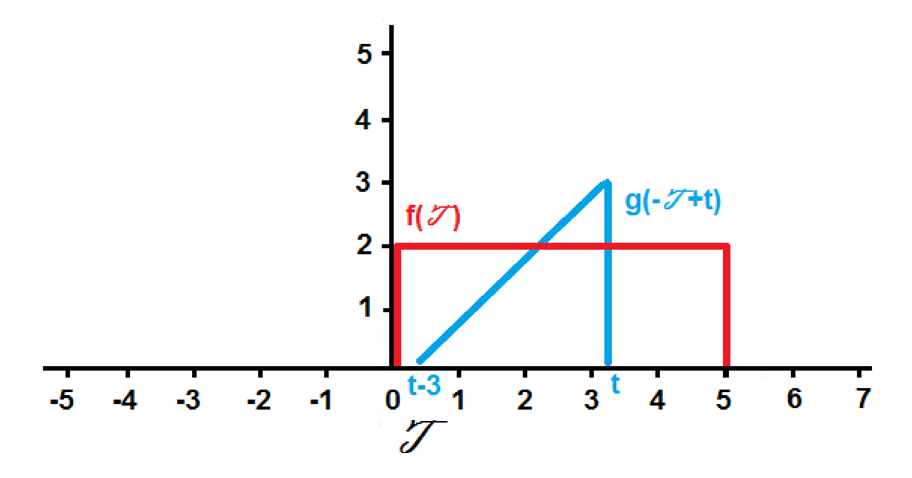
Convolución para t entre 0 y 3

Para este intervalo se cumple que:

Equation:

$$y(t) = \int_{0}^{t} 2(\tau - t + 3)d\tau = -t^2 + 6\text{t} \Rightarrow 0 \leq t \leq 3$$

Una vez las dos funciones se solapan completamente como se observa en la figura 4, la expresión para la convolución será como la descrita en la ecuación , esto ocurre para valores de t situados entre 3 y 5:

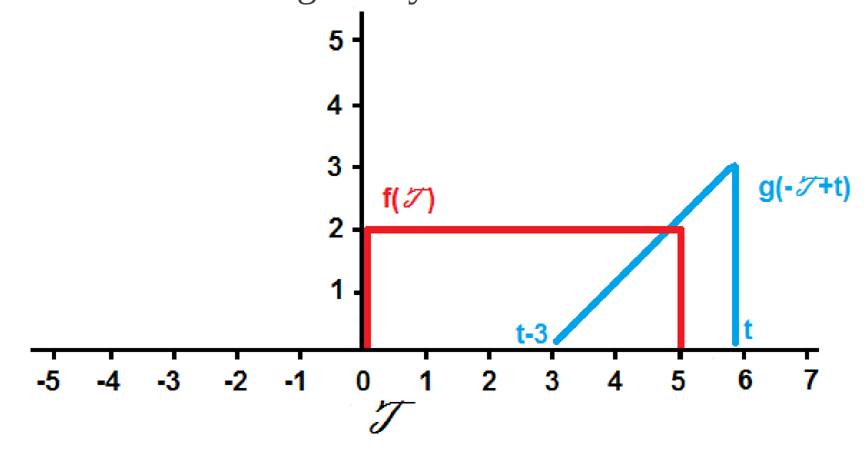


Convolución para t entre 3 y 5

Equation:

$$y(t) = \int_{t-3}^{t} 2(\tau - t + 3)d\tau = 9 \Rightarrow 3 < t \leq 5$$

El siguiente intervalo es el ocurrido para valores de t entre 5 y 8, el mismo se observa en la figura 5 y se describe con la ecuación :

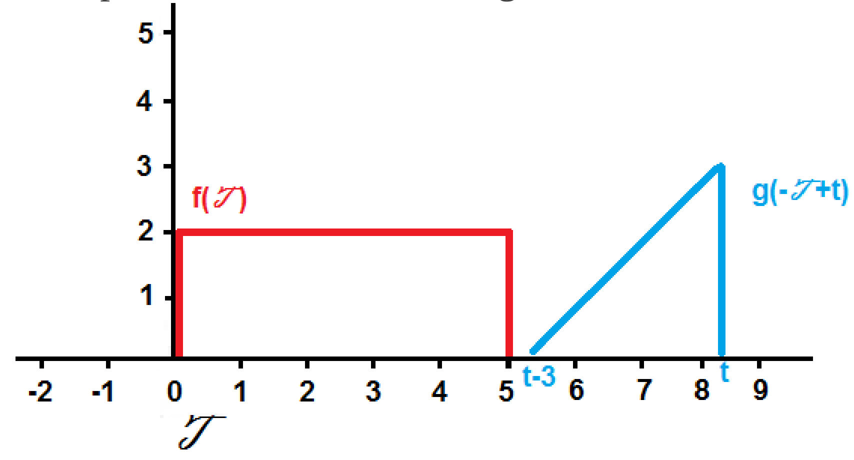


Convolución para t entre 5 y 8

Equation:

$$y(t) = \int_{t-3}^{5} 2(\tau - t + 3)d\tau = (t-8)^2 \Rightarrow 5 < t \leq 8$$

Para valores de t mayores a 8 las funciones no volverán a intersectarse como puede observarse en la figura 6:



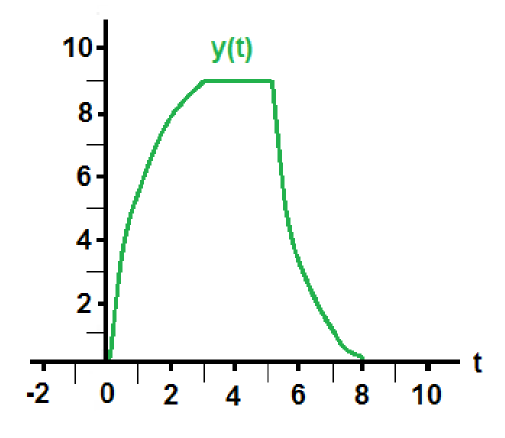
Convolución para t mayor que 8

Como se explicó antes, la convolución vale 0 en los puntos donde las funciones no se intersectan:

Equation:

$$y(t) = 0 \Rightarrow t > 8$$

Con los resultados obtenidos para cada intervalo, mostrados en las ecuaciones anteriores puede construirse la función resultante y(t) = f(t) $*$g(t):



Resultado de la Convolución

Propiedades de la convolución

Propiedad conmutativa:
Equation:

$$f(t) * g(t) = g(t) * f(t)$$

De esta propiedad puede concluirse que es indiferente cuál de las dos funciones será la que se invierta y traslade, y cuál se queda fija.

Propiedad asociativa:
Equation:

$$f(t) * [g(t) * h(t)] = [f(t) * g(t)] * h(t)$$

Propiedad distributiva:
Equation:

$$f(t) * [g(t) + h(t)] = [f(t) * g(t)] + [f(t) * h(t)]$$

Multiplicación por escalar:
Equation:

$$\mathrm{af}(t) * g(t) = f(t) * \mathrm{ag}(t) = a[f(t) * g(t)]$$

Siendo a cualquier número real o complejo.

Derivación:
Equation:

$$\frac{\partial(f(t))}{\partial t} * g(t) = f(t) * \frac{\partial(g(t))}{\partial t} = \frac{\partial[f(t) * g(t)]}{\partial t}$$

Transformada de Fourier de la convolución:
Equation:

$$F[g(t) * h(t)] = F[f(t)] \cdot F[g(t)]$$

Con esta propiedad puede demostrarse la propiedad de la Convolución por delta de Dirac:
Equation:

$$f(t) * \delta(t) = f(t)$$

Si se aplica la transformada de Fourier a la expresión $f(t) * \delta(t)$ se obtendrá el producto de las transformadas; la transformada de Fourier de la función Delta de Dirac es igual a 1, por lo que sólo quedará la transformada de la función $f(t)$. Generalizando la ecuación 15 se obtiene que:
Equation:

$$f(t) * \delta(t - t_1) = f(t - t_1)$$

Equation:

$$f(t - t_2) * \delta(t - t_1) = f(t - t_1 - t_2)$$

Compresión de voz por medio de Transformadas
Transformada de Fourier, Transformada Coseno y Transformada Ondícula aplicadas a la compresión de señales de voz. Se incluye un programa en MATLAB y otro en LabVIIEW, cada uno aplica estas transformadas y la cuantificación para comprimir señales de voz.

Existen diversos métodos para reducir la cantidad de bits que se almacenan o se transmiten a fin de representar una **señal** particular, por ejemplo una señal de voz. Uno de los métodos usados consiste en aplicar alguna transformada como la de **Fourier**, la **Transformada Coseno** o la **Transformada Ondícula** a la señal que se quiere comprimir y reducir los elementos en estos nuevos dominios: magnitud, fase, etc. Por ejemplo, pueden asignarse valores nulos a ciertos elementos de la transformada, normalmente a las que aporten menos información significativa.

Otra forma de comprimir es **cuantificando** los elementos en el dominio transformado y luego antitransformar. Una cuantificación usando 8 bits representa una reconstrucción casi exacta de la señal; puede cuantificarse usando menos bits para ciertas zonas de la transformada (o para toda la transformada) y de esta forma se logra comprimir aún más.

Para comparar la señal original y la señal comprimida se hace uso del error cuadrático medio. El error cuadrático medio entre dos señales w(n), y(n) de K puntos se determina como indica la siguiente expresión:
Equation:

$$\prec \varepsilon^2 \succ = \sum_{K} \frac{\left(w(n) - y(n)\right)^2}{K}$$

Transformada de Fourier

Al aplicar la Transformada de Fourier a una señal en el dominio del tiempo, se observa el comportamiento frecuencial de dicha señal, específicamente, se observan los valores de frecuencia que conforman a la señal. Aplica también para los **sistemas**, si se aplica la Transformada de Fourier a la respuesta impulsiva de un sistema, se obtendrá la respuesta en frecuencia

del mismo, también denominada Función de Transferencia. Al multiplicar la respuesta en frecuencia del sistema con el comportamiento frecuencial de una señal, se obtendrá el comportamiento frecuencial de la señal de salida. La expresión para la transformada de Fourier es la siguiente:

Equation:

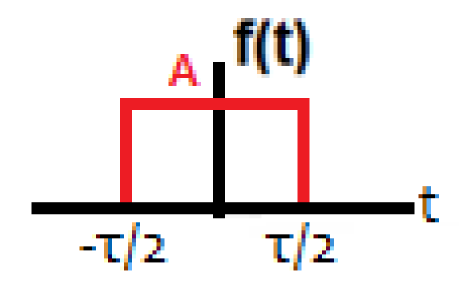
$$X(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j2\pi \mathrm{ft}} \mathrm{dt}$$

Si se tiene el comportamiento frecuencial de una señal, la misma puede recuperarse con una expresión similar:

Equation:

$$x(t) = \int_{-\infty}^{\infty} X(f) \cdot e^{j2\pi \mathrm{ft}} \mathrm{df}$$

Como ejemplo, se determina la transformada de Fourier del pulso cuadrado de la figura 1. Los valores de amplitud (A) y duración (τ) se dejan expresados:



Pulso cuadrado

La función solo está definida entre – τ/2 y τ/2, intervalo para el que toma un valor de A, por lo que la expresión para la transformada de Fourier queda de la siguiente forma:

Equation:

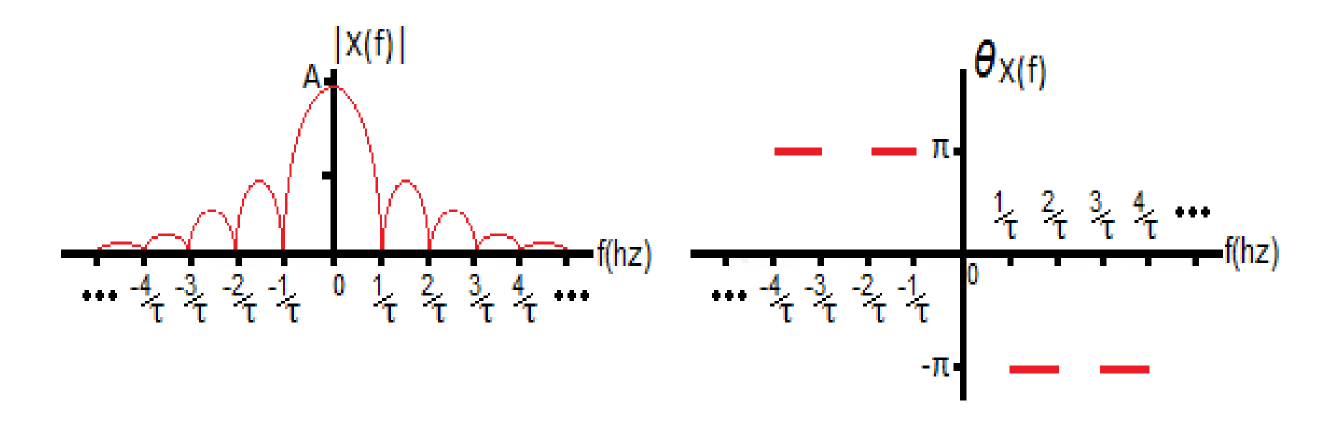
$$X(f) = \int_{-\tau/2}^{\tau/2} A \cdot e^{-j2\pi \mathrm{ft}} \mathrm{dt} = \frac{A}{-2\mathrm{j}\pi \mathrm{f}} \left[e^{-j\pi f\tau} - e^{j\pi f\tau} \right]$$

Simplificando esta expresión queda:

Equation:

$$\frac{A}{-2\mathrm{j}\pi \mathrm{f}} [-\sin \pi f\tau] \cdot 2\mathrm{j} = A\tau \cdot \mathrm{Sinc}(f\tau)$$

El espectro bilateral en magnitud y fase para la señal X(f) se muestra en la figura 2; un valor de fase de π o –π representa valores negativos en la función, los mismos aparecen en el espectro de fase en las zonas donde el Sinc es negativo; en el espectro se debe alternar entre π y –π ya que la fase de la transformada de Fourier es una función impar.



Espectro Bilateral en Magnitud (derecha) y en Fase (izquierda)

Propiedades de la Transformada de Fourier

Linealidad: la Transformada de Fourier cumple con los principios de superposición y multiplicación por constante; si $X_1(f)$ es la transformada de $x_1(t)$ y $X_2(f)$ es la transformada de $x_2(t)$ se cumple que:

Equation:

$$\alpha \cdot x_1(t) + \beta \cdot x_2(t) \xrightarrow{\mathrm{F}} \alpha \cdot X_1(f) + \beta \cdot X_2(f)$$

Traslación en tiempo: si X(f) es la transformada de x(t) se cumple que:

Equation:

$$x(t - t_0) \xrightarrow{\mathrm{F}} X(f) \cdot e^{-j2\pi f \cdot t_0}$$

Traslación en frecuencia: si X(f) es la transformada de x(t) se cumple que:

Equation:

$$x(t) \cdot e^{j2\pi t \cdot f_0} \xrightarrow{\mathrm{F}} X(f - f_0)$$

Esta propiedad se conoce como Teorema de Modulación; en aplicaciones reales, la señal en tiempo se multiplica por la señal senoidal $\cos(2\pi f_0 t)$, la cual es representada por medio de exponenciales, quedando la ecuación de la siguiente forma:

Equation:

$$x(t) \cdot \left[\frac{e^{j2\pi t \cdot f_0} + e^{-j2\pi t \cdot f_0}}{2} \right] \xrightarrow{\mathrm{F}} \frac{1}{2} \left[X(f - f_0) + X(f + f_0) \right]$$

Cambio de escala: si X(f) es la transformada de x(t) se cumple que:

Equation:

$$x(\alpha t) \xrightarrow{\mathrm{F}} \frac{1}{\mid \alpha \mid} X(f/\alpha)$$

Teorema de Rayleigh: si X(f) es la transformada de x(t) se cumple que:
Equation:

$$\text{Energía} = \int_{-\infty}^{\infty} \mid x(t) \mid^{2} \text{dt} = \int_{-\infty}^{\infty} \mid X(f) \mid^{2} \text{df}$$

Transformada de Fourier de la **Convolución**: si $X_1(f)$ es la transformada de $x_1(t)$ se y $X_2(f)$ es la transformada de $x_2(t)$ cumple que:
Equation:

$$x_1(t) * x_2(t) \overset{\text{F}}{\rightarrow} X_1(f) \cdot X_2(f)$$

Transformada Discreta de Fourier

La Transformada de Fourier aplica también para señales discretas, con la condición de que las mismas tengan una duración finita. La expresión para la Transformada Discreta de Fourier de una señal discreta x[n] de longitud N es la siguiente:
Equation:

$$X[k] = \sum_{n=0}^{N-1} x_n \cdot e^{-j\frac{2\pi}{N}\text{kn}}$$

Transformada Discreta Coseno

La transformada Discreta Coseno (DCT) está relacionada a la Transformada Discreta de Fourier. Se calcula como:
Equation:

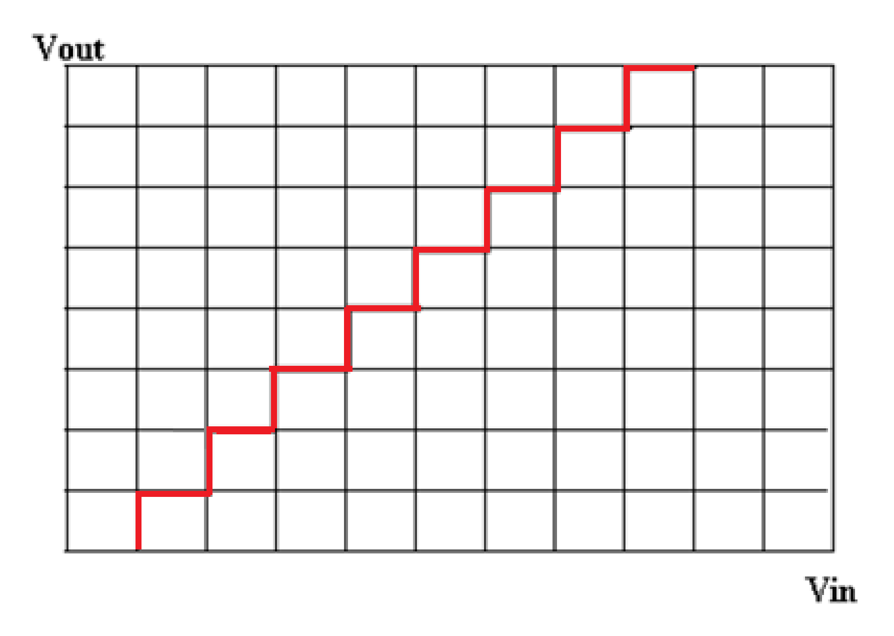
$$X[k] = A\sum_{n=0}^{N-1} x(n)\cos\left(\frac{\pi k(2\mathrm{n}+1)}{2\mathrm{N}}\right)$$

$$A = \begin{matrix} \frac{1}{\sqrt{N}} k = 0 \\ \frac{2}{\sqrt{N}} k = 1{,}2 \ldots N - 1 \end{matrix}$$

En la fórmula anterior N es la longitud de x. La ventaja de la DCT es que compacta la información alrededor del origen de coordenadas. Por eso es usada en algunos algoritmos de compresión como el JPEG, debido a la compactación de energía es posible reconstruir una señal usando solo unos pocos coeficientes de la DCT.

Cuantificación uniforme y no uniforme de señales de voz

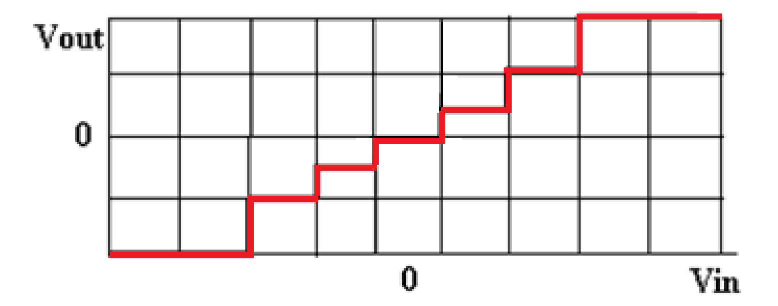
Cuando una señal analógica quiere digitalizarse deben realizarse varios procesos entre ellos están el muestreo y la cuantificación. El muestreo consiste en tomar muestras de la señal periódicamente; el tiempo entre muestra y muestra denominado ts. Esto discretiza la señal en el dominio del tiempo. El siguiente proceso es la cuantificación en donde la señal ya muestreada es pasada por un sistema que presenta la siguiente característica:



Cuantificación uniforme

Es decir, se observa cada muestra y se ubica en que rango de voltaje se encuentra; dependiendo de esta se le asigna un voltaje de salida. Es decir, la señal de entrada tiene infinitos valores de voltaje posibles, mientras la señal de salida tiene un número finito de voltajes posibles. Por ejemplo, si se divide el rango de entrada en 256 intervalos y a cada intervalo se le asigna un determinado voltaje de salida, a la salida se tendrán solo 256 voltajes distintos posibles; en este caso particular se necesitarían 8 bits para representarlos. Este tipo de cuantificador se le llama Cuantificador Uniforme.

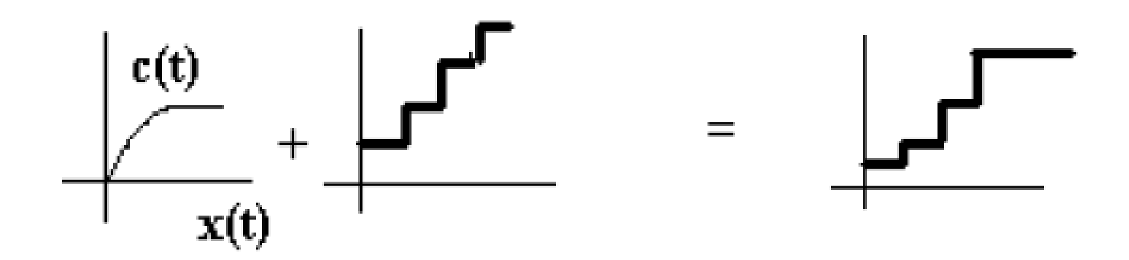
Cuando la distribución probabilística de x(t) no es uniforme sino que tiene más bien preferencia por una cierta zona de voltaje, como el caso de las señales de voz, conviene usar cuantificadores no uniformes, es decir uno que tenga pasos más estrechos en aquellas zonas de voltaje más frecuentes y pasos más gruesos en zonas menos probables. Gráficamente:



Cuantificación no uniforme

Por ejemplo, el cuantificador mostrado convendría usarlo cuando la señal tiene preferencia de ocurrencia en los voltajes alrededor de cero; En el caso de señales de voz esto es en efecto lo que ocurre.

Este tipo de cuantificación se le llama cuantificación no-uniforme y puede ser lograda haciendo pasar la señal por un sistema llamado compansor el cual expande los valores de bajo voltaje y comprime los de alto voltaje y posteriormente pasar esta señal por un cuantificador no-uniforme, tal y como se ilustra a continuación:



Compansor + Cuantificación uniforme = Cuantificación no uniforme

Por supuesto que en el receptor hay que proveer de un sistema que haga el efecto inverso al de c(t) vs. x(t)

Autoevaluación

Exercise:

Problem:

Si se tiene un sistema que comprime eliminando componentes de la Transformada de Fourier desde las frecuencias altas como si se tratase de un filtrado pasabajos, ¿las voces que se podrán comprimir más son las masculinas o las femeninas? ¿Por qué?

Solution:

Si se elimina la misma cantidad de componentes de la Transformada de Fourier desde las frecuencias altas para una voz masculina y una voz femenina, la voz masculina tendrá una mejor calidad con respecto a la original ya que para ésta tienen más relevancia las componentes de menor frecuencia

Exercise:

Problem:

Si una señal cuya Transformada Coseno que se cuantifica originalmente de forma uniforme con 8 bits, se comprime utilizando sólo 7 bits para la cuantificación, ¿de cuánto será la tasa de compresión?

Solution:

La cantidad de bits total de la transformada es su longitud N por la cantidad de bits (8 originalmente). La relación entre la longitud de la señal comprimida y la señal original será de 7N/8N=0.875 → 87.5%, por lo que la tasa de compresión es de 12.5%. El resultado de la tasa de compresión es independiente de la transformada que es usada.

Exercise:

Problem:

Si una señal se comprime aplicando la Transformada Ondícula de profundidad 3 y cuantificando dicha transformada originalmente con 8 bits, ¿De cuánto será la tasa de compresión si se elimina el detalle de mayor frecuencia y si se utilizan 4 bits para cuantificar?

Solution:

Al eliminar el detalle de mayor frecuencia, la transformada queda con la mitad de la longitud; al cuantificar con 4 bits, la cantidad de bits se reduce a la mitad, quedando la relación comprimida/original: $(4N/2)/8N=0.25 \rightarrow$ 25%, por lo que la tasa de compresión es de 75%.

Simuladores

ESTE VINCULO contiene una carpeta con un programa realizado en **MATLAB** que comprime señales de voz por medio de la Transformada Ondícula, La Transformada Coseno y la Transformada de Fourier; para esta última, la compresión se hace eliminando componentes desde las altas frecuencias. Con cada transformada se hace cuantificación uniforme. La carpeta incluye el .m y todos los archivos necesarios para su funcionamiento, si se elimina o renombra alguno de estos archivos, el programa podría no funcionar correctamente. La figura 6 contiene un video explicativo acerca del uso del programa.

Compresión de voz en MATLAB

[missing_resource: http://www.youtube.com/v/bxoaLAckAK8?fs=1&hl=es_ES]

Video explicativo de la utilización del programa realizado en MATLAB

Puede obtenerse también un programa realizado en **LabVIEW** del mismo estilo y acerca del mismo tema por medio de ESTE VINCULO. La carpeta

incluye el .vi y todos los archivos necesarios para su funcionamiento. Igualmente, si se elimina o renombra alguno de estos archivos, el programa podría no funcionar correctamente. La figura 7 contiene un video explicativo acerca del uso del programa

Compresión de voz en LabVIEW

[missing_resource: http://www.youtube.com/v/qLflZpOmD2M?fs=1&hl=es_ES]

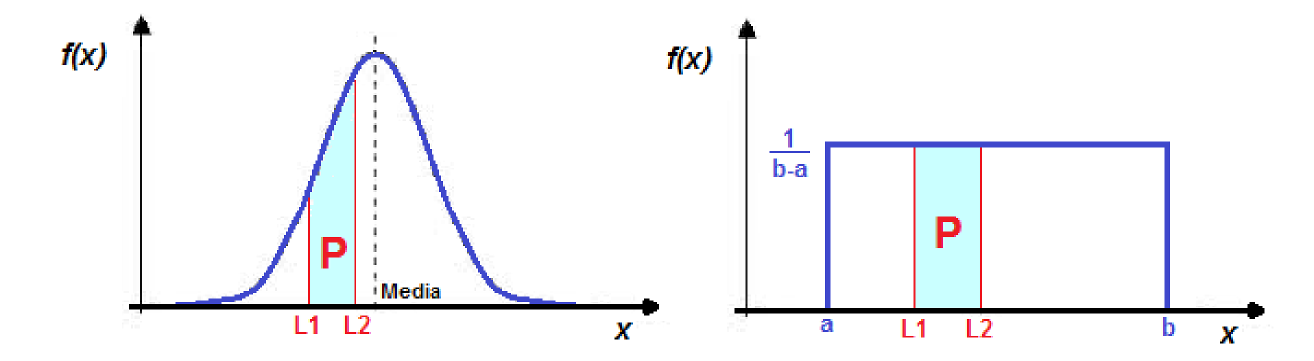
Video explicativo de la utilización del programa realizado en LabVIEW

Procesos Aleatorios
Concepto de proceso aleatorio, procesos estacionarios y ergódicos.

Un Proceso Aleatorio se define como el conjunto de **señales** provenientes de realizar un determinado experimento o de un evento de la naturaleza. La naturaleza aleatoria del experimento proviene del desconocimiento de cuál de las señales se obtendrá al realizar el experimento. Para caracterizar los Procesos Aleatorios se definen diversas variables aleatorias como la secuencia de valores de las diversas señales evaluadas en tiempos específicos. Así se puede hablar de x(t1), x(t2), etc. Procesos Aleatorios pueden ser continuos o discretos. Los casos especiales para Procesos Aleatorios mayormente utilizados en el ámbito de las comunicaciones son los *Procesos Estacionarios* y los *Procesos Ergódicos.*

Función de Densidad de Probabilidades

La Función de Densidad de Probabilidades es una función que, al integrarse entre un límite inferior (L1) y un límite superior (L2), indica la probabilidad de que la variable aleatoria tome valores entre L1 y L2; el área total definida por la función de densidad de probabilidades es igual a 1. Existen varios tipos de distribución, como uniforme, gaussiana, exponencial, entre otras. La figura 1 muestra la función de densidad de probabilidades para una distribución gaussiana y una distribución uniforme respectivamente, el área pintada en azul claro representa el valor de la probabilidad de que la variable tome valores entre L1 y L2:



Función de Densidad de Probabilidades Gaussiana y Uniforme

El concepto de función de densidad de probabilidades puede generalizarse a más de una variable, convirtiéndose en una función n-dimensional denominada Función de Densidad de Probabilidades Conjunta.

Procesos Estacionarios

Si la función de densidad de probabilidades aplicada a una señal aleatoria en cierto instante es igual si la misma se desplaza cualquier valor de tiempo, se dice que representa un proceso estacionario de primer orden, resumiendo:
Equation:

$$\mathrm{fdp}(x(t)) = \mathrm{fdp}(x(t+\tau))$$

Tomándose en cuenta dos variables aleatorias de un mismo proceso: $x(t_1)$ y $x(t_2)$, si la función de densidad de probabilidades conjunta aplicada a ambas variables aleatorias es igual si para un desplazamiento de tiempo cualquiera, se dice que el proceso es estacionario de segundo orden:
Equation:

$$\mathrm{fdp}(x_1(t),x_2(t)) = \mathrm{fdp}(x_1(t+\tau),x_2(t+\tau))$$

En general, se dice que un proceso es estacionario de orden N si se cumple que:
Equation:

$$\mathrm{fdp}(x_1(t),x_2(t)...x_N(t)) = \mathrm{fdp}(x_1(t+\tau),x_2(t+\tau)...x_N(t+\tau))$$

Cualquier proceso estacionario de cierto orden, será estacionario en órdenes inferiores.

Procesos Ergódicos

Un proceso aleatorio es ergódico respecto al primer momento si el promedio estadístico (o valor esperado E[x(t)]) y el promedio temporal (<x(t)>) coinciden; resumiendo:
Equation:

$$\int_{-\infty}^{\infty} x(t)\mathrm{fdp}(x(t))\mathrm{dx}(t) = \lim_{T\to\infty} \frac{1}{T} \int_{T} x(t)d(t)$$

Generalizando, se define la ergodicidad en orden N:
Equation:

$$\int_{-\infty}^{\infty} x^N(t)\mathrm{fdp}(x(t))\mathrm{dx}(t) = \lim_{T\to\infty} \frac{1}{T} \int_{T} x^N(t)d(t)$$

Cualquier proceso ergódico de cierto orden, es estacionario en ese mismo orden, además será ergódico en órdenes inferiores. Para procesos ergódicos de segundo orden se cumple que:

- El nivel DC de la señal es igual al 1er momento: $\langle x(t)\rangle$
- La potencia promedio total de la señal es igual al 2do momento: $\langle x^2(t)\rangle$
- La potencia AC de la señal se conoce como varianza: $\langle x^2(t)\rangle - \langle x(t)\rangle^2$

Autocorrelación

La Autocorrelación es una función que indica la relación que tiene el valor que toma una señal en un momento específico con sus vecinos temporales. El concepto de Autocorrelación se aplica para señales determinísticas y aleatorias aunque para estas últimas es una herramienta insustituible si el Proceso es Ergódico; la expresión para la misma corresponde con el valor esperado de la multiplicación de la señal en un tiempo t_1 por la misma señal en un tiempo t_2:

Equation:

$$\mathfrak{R}_x = E[x(t_1)x(t_2)] = E[x(t)x(t+\tau)]$$

La variable τ de la función de autocorrelación hace referencia a la diferencia entre los instantes de tiempo involucrados t1 y t2. Si el proceso es ergódico, puede sustituirse la expresión para el valor esperado por la expresión para el promedio temporal como indica la ecuación 4, quedando así una expresión determinística:

Equation:

$$\mathfrak{R}_x(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_T x(t)x(t-\tau)d(t)$$

Densidad Espectral de Potencia (DEP):

La DEP de una señal indica cómo está distribuida la potencia de la señal en función de la frecuencia. Para Procesos Ergódicos corresponde con la **Transformada de Fourier** de la función de autocorrelación y su área coincide con la potencia promedio total de la señal, y coincide a su vez la autocorrelación en τ=0.

Modulaciones AM-DSB-SSB, Repetidoras y Ruido Pasabanda
Se explican diversos tipos de modulación en amplitud, se explica el efecto del canal para estos tipos de modulación y se ilustra el uso de repetidoras. Se incluye un programa en MATLAB y otro en LabVIEW, cada uno acerca de un sistema que incluye distintos tipos de modulación, repetidoras intermedias y caracterización del ruido

Todo sistema de transmisión tiene un transmisor, un canal y un receptor. Cuando se desea compartir un canal conviene utilizar algún tipo de modulación. El canal tiene ciertos efectos sobre el **mensaje** transmitido, el mismo atenúa la señal, añade ruido blanco, entre otros efectos, los cuales se harán más notorios mientras mayor sea la distancia entre transmisor y receptor. Es por esto que en estos **sistemas** se hace el uso de Repetidoras, las cuales amplifican y hacen regeneraciones de la señal en puntos intermedios del trayecto, logrando así un mayor alcance para la comunicación. En este módulo se hace referencia a los esquemas de modulación AM (Modulación de Amplitud), DSB (Doble Banda Lateral) y SSB (Banda Lateral Única), se exponen las características del Ruido que afecta a la comunicación en su paso por el canal inalámbrico, y por último se explica el funcionamiento de las Repetidoras Analógicas.

Modulación AM, DSB y SSB

Si se conoce como x(t) al mensaje o señal original, las fórmulas para cada modulación serán las siguientes:
Equation:

$$x_{\mathrm{AM}}(t) = A_c(1 + \mathrm{mx}(t))\mathrm{Cos}\omega_c t$$

Equation:

$$x_{\mathrm{DSB}}(t) = A_c x(t)\mathrm{Cos}\omega_c t$$

Equation:

$$x_{\mathrm{SSB}}(t) = 0.5\mathrm{A}_c x(t)\mathrm{Cos}\omega_c t \pm 0.5\mathrm{A}_c \hat{x}(t)\mathrm{Sen}\omega_c t$$

En las fórmulas anteriores:

- m es el índice de modulación, el mismo está comprendido entre 0 y 1.
- La señal senoidal *Cos* $\omega_c t$ es la portadora.
- *Ac* es la amplitud de la portadora.
- ω_c equivale a $2\pi f_c$ donde f_c es la frecuencia de la portadora.
- $\hat{x}(t)$ es la transformada de Hilbert de x(t) la cual representa lo siguiente, en los dominios de tiempo y frecuencia:

Equation:

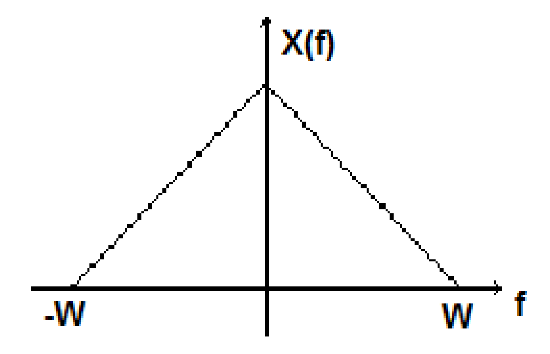
$$\hat{x}(t) = \frac{1}{\pi t} * x(t)$$

Equation:

$$\hat{X}(f) = -j\text{sgn}(f)X(f)$$

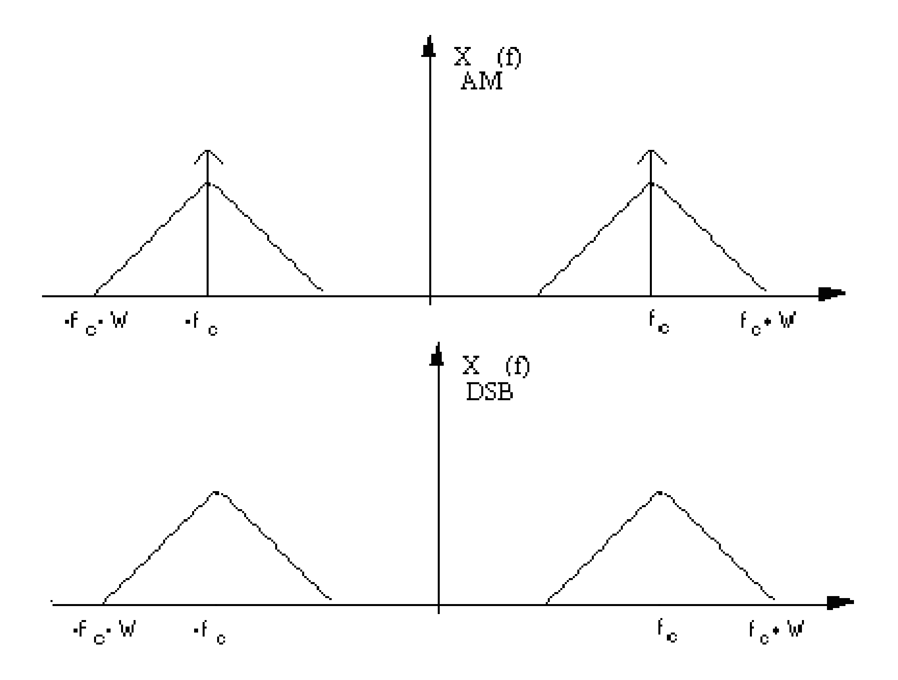
Es decir la transformada de Hilbert puede verse como un desfasador de -90°.

Para el mejor entendimiento de cómo sería el comportamiento en frecuencia de este sistema de modulación supóngase que X(f) (**representación en frecuencia** del mensaje x(t)) luce como se muestra en la figura 1:



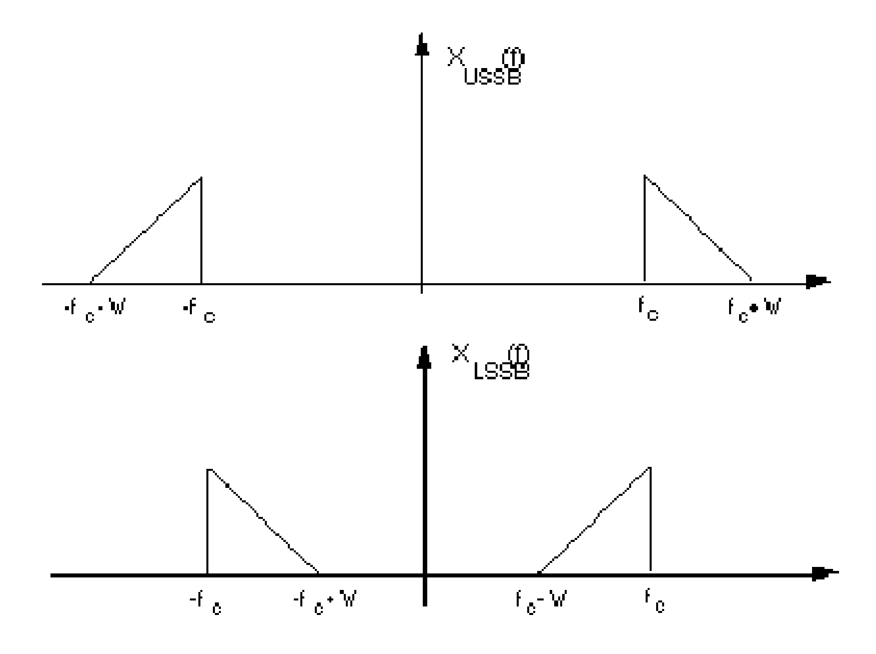
Mensaje original en frecuencia

Las señales moduladas en AM y DSB (figura 2) tendrían un espectro parecido al del mensaje solo que trasladado alrededor de la frecuencia de portadora f_c. Además en AM aparece la portadora en f_c en forma de delta (recordándose que el comportamiento en frecuencia de una señal senoidal luce como una delta).



Espectro de modulaciones AM y DSB

En cambio, en SSB, dependiendo del signo elegido en la fórmula temporal de la señal modulada, lucirá en frecuencia de la siguiente forma:



Espectro de modulaciones SSB

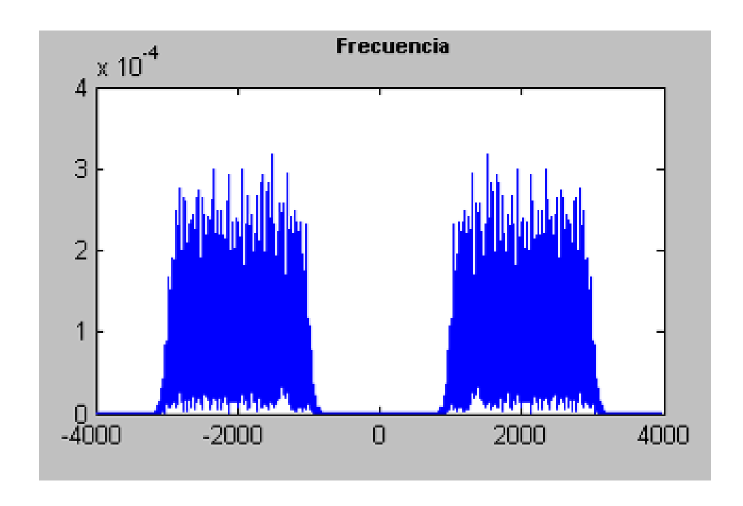
Si en la fórmula original se toma el signo (-) se tendrá USSB (Upper Single Side Band) es decir se toma la banda superior del espectro del mensaje original. Si se toma el signo (+) entonces se tendrá LSSB (Lower Single Side Band) es decir se toma la banda inferior.

Para rescatar cada uno de los mensajes existen varias técnicas diferentes entre las que se encuentra lo que se conoce como detector síncrono, que no es más que un multiplicador por una sinusoide de frecuencia igual a la de la portadora, seguido de un filtro pasabajos de frecuencia de corte igual a la del mensaje y de un bloqueador de DC. Otra técnica algo más compleja pero más efectiva es el **Receptor Superheterodino**, el cual puede o no utilizar el detector coherente.

Caracterización del Ruido Pasabanda

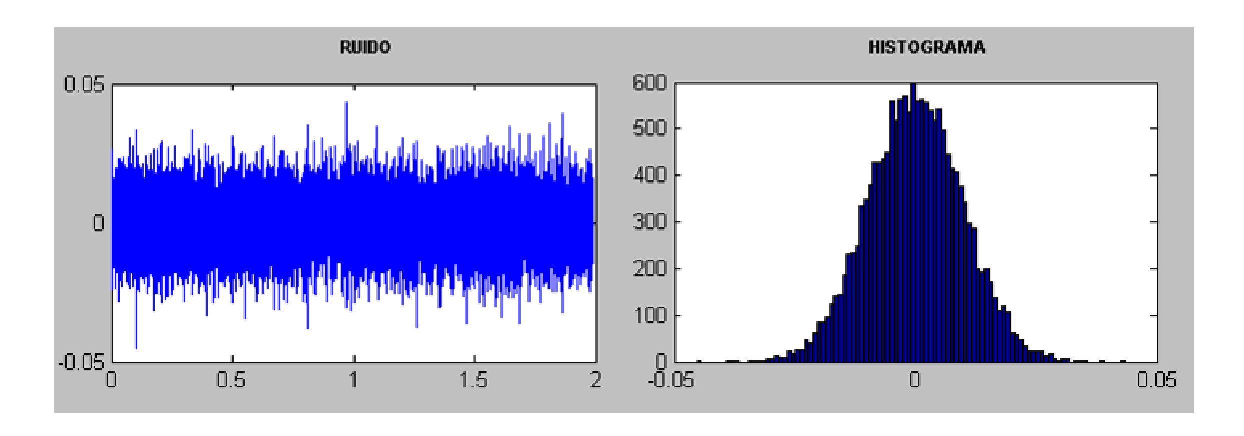
Una vez modulada la señal en el canal se contaminará con ruido blanco, gaussiano y con una Densidad Espectral de Potencia que se modela como constante (0.5*n*) para todo valor de frecuencia. La suma de señal modulada

y el ruido blanco gaussiano debe ser pasada por un filtro denominado filtro RF, el cual es de tipo pasabanda que estará centrado en f_c y tendrá como ancho de banda el doble del mensaje en caso de que la modulación sea AM o DSB, tendrá un ancho de banda igual al mensaje y su frecuencia de corte superior será f_c en caso de que la modulación sea LSSB, o tendrá un ancho de banda igual al mensaje y una frecuencia de corte inferior igual a f_c en caso de que la modulación sea USSB. En la figura 4 se muestra la Densidad Espectral de Potencia del ruido pasado por un filtro RF centrado en 2000Hz y con un ancho de banda de 1000Hz:



Ruido en frecuencia (filtrado)

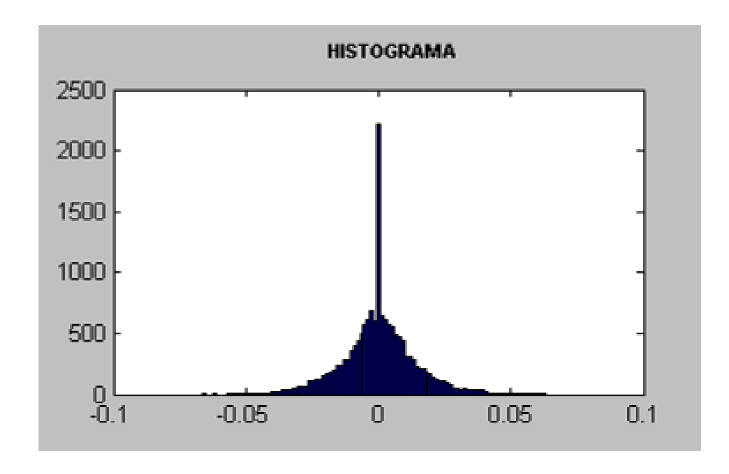
El ruido suele ser caracterizado también por medio del histograma (figura 5). El histograma es una representación en barras que indica el número de veces que una función tuvo valores en cierta cantidad de intervalos. A cada intervalo le corresponden los valores ocupados por la barra correspondiente.



Ruido en tiempo (izquierda) e histograma del ruido (derecha)

El histograma está centrado y presenta un mayor número de muestras alrededor de un valor que corresponde a la media del ruido (casi 600), por lo que se comprueba que el ruido tiene media 0; se observa también que los valores se repiten un menor número de veces mientras más se alejan de la media, y que la forma dada a la gráfica por este decrecimiento tiene la forma de una campana de Gauss, por lo que se dice que el ruido es gaussiano.

En la figura 6 se observa el histograma de la señal demodulada pero no pasada por el filtro pasabajos aun:

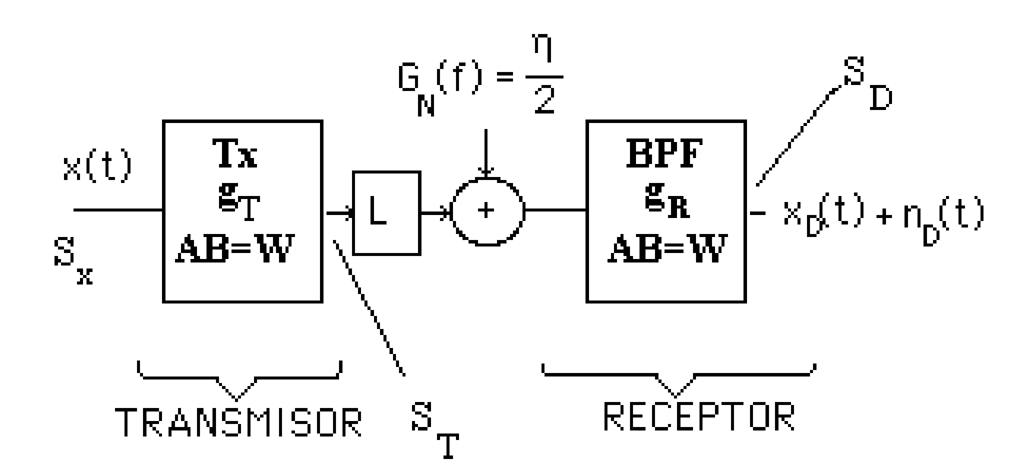


Ruido demodulado sin filtrar

En este caso el histograma muestra un mayor número de muestras cercanas a 0, esto se debe a que el ruido ha sido multiplicado por una señal senoidal que cíclicamente toma valores cercanos a 0, por lo cual se incrementa la cantidad de veces que el valor del ruido pertenece a dicho intervalo. Este sencillo programa realizado en **LabVIEW** hace una demostración directa de este comportamiento.

Repetidoras Analógicas

En la figura 7 se muestra un esquema que incluye el transmisor, el efecto del canal y la primera etapa del receptor.



Transmisor, canal y receptor

Se tiene a la salida del transmisor una señal modulada, que puede ser similar a la mostrada en figuras anteriores, con un ancho de banda W y potencia S_T. Luego el canal produce una atenuación de potencia L y el filtro pasabanda del receptor una amplificación g_R. Así, la potencia de señal a la salida de este filtro será:

Equation:

$$S_D = \frac{g_R \cdot S_T}{L}$$

Esto es asumiendo que el filtro del receptor tiene un ancho de banda apropiado (W) para que pase todo el contenido de señal. Por su parte el ruido sólo se afecta por g_R. La DEP del ruido de entrada es constante (ruido blanco) y con valor η/2; al pasar por el filtro, la DEP queda confinada entre los valores que limitan en ancho de banda W de tal forma que al integrar la DEP para obtener la potencia del ruido a la salida queda:
Equation:

$$N_D = g_R \cdot \eta \cdot W$$

Finalmente, se calcula la relación señal a ruido (S/N) a la salida del filtro dividiendo la potencia de la señal entre la potencia del ruido en este punto:
Equation:

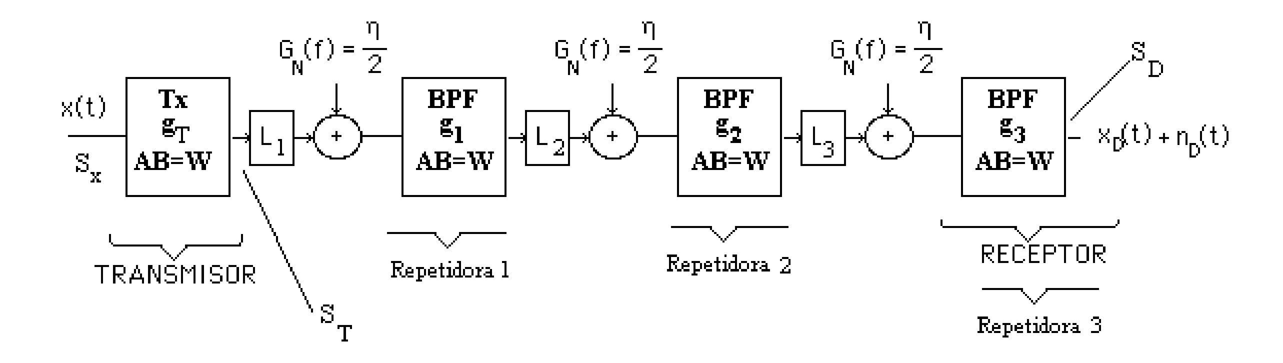
$$\left(\frac{S}{N}\right)_D = \frac{g_R \cdot S_T}{L \cdot \eta \cdot g_R \cdot W} = \frac{S_T}{L \cdot \eta \cdot W}$$

Se observa que la relación señal a ruido recibida aumenta cuando la potencia de señal S_T aumenta, cuando la atenuación L disminuye o cuando La potencia del ruido (η o W) disminuye. Además se nota que la relación señal a ruido, y por ende la calidad de la transmisión:

- No depende de la ganancia del receptor.
- Es inversamente proporcional al ancho de banda del filtro.
- Es inversamente proporcional a la atenuación que produce el canal.

Una manera de mejorar la relación señal a ruido es colocando estaciones repetidoras equi-espaciadas en zonas intermedias del trayecto de transmisión. El diagrama de la figura 8 ilustra el uso de estas estaciones para un caso en el que se cuenta con 2 repetidoras intermedias más el

receptor; el receptor se cuenta también como una repetidora, ya que el mismo también amplifica la señal de la misma forma que lo hacen las demás:



Sistema con tres repetidoras (dos repetidoras intermedias + receptor)

Normalmente la ganancia de cada repetidor compensa la pérdida del trayecto (L_k=g_k); de esta forma, la potencia de señal se mantiene a la salida del sistema (S_T = $S_{D)}$. Se asumirá que el ruido que ingresa al sistema en cualquier trayecto es el mismo y que las pérdidas parciales son iguales (L_1= L_2 = L_3)

Por su parte, la potencia de ruido a la salida de la primera repetidora será:
Equation:

$$N_{D1} = g_1 \cdot \eta \cdot W = L_1 \cdot \eta \cdot W$$

La potencia de este ruido al final de las m repetidoras queda igual ya que cada pérdida de canal será compensada por la ganancia de cada repetidora. Sin embargo se irán sumando contribuciones idénticas de ruido, tantas como repetidoras existan. Al final para m repetidoras (incluyendo al receptor):
Equation:

$$N_D = m \cdot L_1 \cdot \eta \cdot W$$

Finalmente
Equation:

$$\left(\frac{S}{N}\right)_D = \frac{g_T \cdot S_x}{m \cdot L_1 \cdot \eta \cdot W}$$

Para comparar con la relación señal a ruido sin repetidoras se sustituyen en esta ecuación los valores de la ecuación 8 obteniéndose:
Equation:

$$\left(\frac{S}{N}\right)_{D_{(\text{conRe}p.)}} = \frac{L}{\text{mL}_1}\left(\frac{S}{N}\right)_{D_{(\text{sinRe}p.)}}$$

Interesa también esta comparación en decibelios, para esto se aplica 10Log a ambos lados de la ecuación obteniéndose:
Equation:

$$\left(\frac{S}{N}\right)_{D_{(\text{conRe}p.)[\text{dB}]}} = 10\text{Log}\left(\frac{L}{\text{mL}_1}\right) + \left(\frac{S}{N}\right)_{D_{(\text{sinRe}p.)[\text{dB}]}}$$

Para ilustrar esto, considere un sistema basado en un medio inalámbrico con una pérdida total de 60 dB ($L=10^{60/10}$) en lineal, si se coloca una repetidora intermedia, existirán dos trayectos con la mitad de la atenuación, es decir 30dB ($L_1=10^{30/10}$), el receptor se cuenta como repetidora, por tanto m=2. La ganancia en este caso será de:
Equation:

$$10\text{Log}\left(\frac{10^6}{2 \cdot 10^3}\right) = 10\text{Log}(500) = 27\text{dB}$$

Esto significa que la relación señal a ruido lineal se multiplica por 500 o que la relación señal a ruido en dB se incrementa en 27dB produciéndose una mejora considerable en la calidad de la transmisión. Si en vez de agregar una sola, se agregan dos repetidoras intermedias, existirán 3 trayectos con una atenuación de 20dB ($L_1=10^{20/10}=100$), para este caso m es igual a 3 y la ganancia aumentará con respecto al caso anterior:

Equation:

$$10\mathrm{Log}\left(\frac{10^6}{3\cdot 100}\right) = 10\mathrm{Log}(3333.33) = 35.23\mathrm{dB}$$

Para toda transmisión existe un número M máximo de repetidoras, esto significa que la ganancia aportada por M+1 repetidoras es menor que la ganancia aportada por M repetidoras, para este número se cumple que: $(M+1)L_{1[M+1]} > M\ L_{1[M]}$

Autoevaluación

Exercise:

Problem:

¿Puede recuperarse una señal SSB con el filtro pasabanda correspondiente al de una señal DSB a la misma frecuencia?

Solution:

Puede recuperarse, pero debe tenerse en cuenta que esto empobrece la calidad del sistema, ya que la relación señal a ruido es inversamente proporcional al ancho de banda del filtro, el cual se está duplicando innecesariamente en este caso.

Exercise:

Problem:

Observado lo ocurrido con los valores cercanos a 0 en el histograma de la figura 6, ¿cómo varia este resultado con respecto a la frecuencia de portadora?

Solution:

En el mundo continuo, si la frecuencia de portadora aumenta, habrá un mayor número de pasos por 0 en el ruido multiplicado por la portadora, por lo que la barra central en el histograma será mucho mayor y las demás barras disminuirán de tamaño. En el mundo discreto (el caso de MATLAB y LabVIEW) la barra central será mayor si la relación entre la frecuencia de muestreo y la frecuencia de modulación es un numero racional o entero.

Exercise:

Problem:

Una relación S/N a la salida del filtro pasabanda mayor para una señal AM que para una señal DSB, ambas con una misma amplitud de portadora indica una mayor calidad de la modulación AM, ¿Verdadero o Falso?

Solution:

Falso, la señal AM tiene una mayor relación S/N debido a que la portadora está presente, y la misma no forma parte del mensaje.

Exercise:

Problem:

Si la ganancia de cada repetidora fuese mayor que la pérdida por trayecto en vez de ser igual, ¿cómo cambiaría la relación señal a ruido?

Solution:

La relación S/N permanecería igual, ya que la ganancia de la repetidora amplifica tanto a la señal como al ruido.

Exercise:

Problem:

Entre una señal SSB y una señal DSB transmitidas a la misma potencia, ¿Cuál tiene mayor inmunidad al ruido a la salida del filtro pasabanda?

Solution:

Si se transmite una señal SSB con la misma potencia con la que se transmite una señal DSB, la primera tendrá una mayor inmunidad al ruido a la salida del filtro pasabanda, ya que se mantiene la potencia de la señal pero la potencia del ruido se reduce a la mitad al reducirse el ancho de banda. Especificando un poco más, la S/N se multiplica por dos o lo que es lo mismo, la $S/N_{[dB]}$ se incrementa en 3dB. (Para duplicar la potencia de transmisión, la amplitud de la portadora debe ser multiplicada por $\sqrt{2} \approx 1.41$)

Simuladores

ESTE VINCULO contiene una carpeta con un programa realizado en **MATLAB** que simula un Sistema AM-DSB-SSB con Repetidoras. La carpeta incluye el .m y todos los archivos necesarios para su funcionamiento, si se elimina o renombra alguno de estos archivos, el programa podría no funcionar correctamente. La figura 9 contiene un video explicativo acerca del uso del programa.

Sistema de Transmisión en MATLAB

[missing_resource: http://www.youtube.com/v/jhiX7k4Nkcg?fs=1&hl=es_ES]

Video explicativo de la utilización del programa realizado en MATLAB

Puede obtenerse también un programa realizado en **LabVIEW** acerca del mismo tema por medio de ESTE VINCULO. La carpeta incluye el .vi y todos los archivos necesarios para su funcionamiento. Igualmente, si se elimina o renombra alguno de estos archivos, el programa podría no funcionar correctamente. La figura 10 contiene un video explicativo acerca del uso del programa.

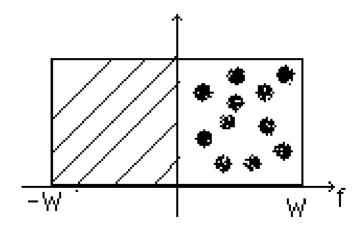
Sistema de Transmisión en LabVIEW

[missing_resource: http://www.youtube.com/v/vrkpjK-8WJU?fs=1&hl=es_ES]

Video explicativo de la utilización del programa realizado en LabVIEW

Transmisión de señales DSB en cuadratura
Envio de dos señales por un mismo canal y a la misma frecuencia. Incluye un programa realizado en MATLAB y otro realizado en LabVIEW acerca de este tema, además de videos explicativos de los mismos.

Se tiene un **mensaje** x(t) (voz por ejemplo) con una **expresión en frecuencia** de X(f) cuya ocupación espectral está entre [-W,W], como se observa a continuación:



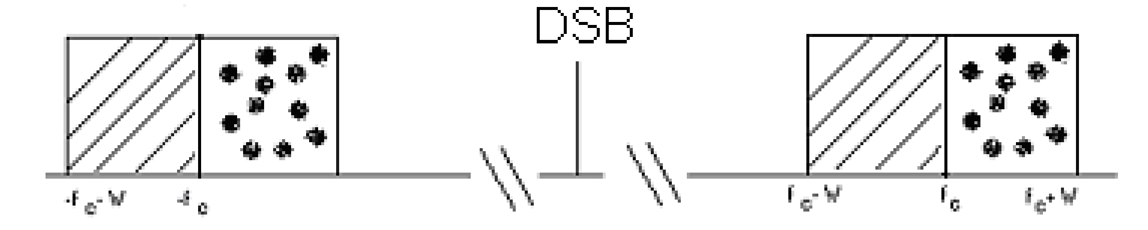
Mensaje en frecuencia.

Para enviar este mensaje por el aire habría que trasladarlo a una frecuencia más alta para poder compartir el canal y además para que la antena sea de una dimensión razonable. Esto podría lograrse usando una **modulación en Doble Banda Lateral (DSB)**, la cual se logra multiplicando el mensaje por una señal sinusoidal, denominada portadora. La señal DSB tiene una expresión como la mostrada en la ecuación 1:

Equation:

$$A_c x(t) \mathrm{Cos}(\omega_c t)$$

El espectroresultante luciría de la siguiente forma:



Señal modulada en DSB.

Si en vez de multiplicar por Coseno, multiplica por Seno, el mensaje se traslada al mismo sitio y ocupa el mismo ancho de banda.

Disponiéndose de un canal en frecuencia centrado en fc y de ancho de banda B (fc>>B) y requiriéndose transmitir 2 señales que, alrededor de fc, ocuparían, cada una de ellas, toda la banda de ancho B, se puede utilizar la opción de enviarlas en cuadratura, es decir, ambas señales x1(t) y x2(t) se envían por un mismo canal modulándose cada una en DSB, esto se logra utilizando una portadora de sen(ω_ct) para x1(t) y una portadora de cos(ω_ct) para x2(t) y sumando las señales obtenidas, dando como resultado la siguiente expresión:

Equation:

$$y(t) = x_1(t)\mathrm{Sen}(\omega_c t) + x_2(t)\mathrm{Cos}(\omega_c t)$$

Esto se denomina modulación en cuadratura. La señal Y(t) ocupará el ancho de banda de cada una de ellas individualmente (no el doble) y aunque están en la misma banda de frecuencias podrán separarse si en el receptor se demodula con ambas portadoras por separado, tal y como lo indican las siguientes ecuaciones:

Equation:

$$y'_1(t) = [x_1(t)\mathrm{Sen}(\omega_c t) + x_2(t)\mathrm{Cos}(\omega_c t)]\mathrm{Sen}(\omega_c t)$$

Equation:

$$y'_1(t) = x_1(t)\mathrm{Sen}^2(\omega_c t) + x_2(t)\mathrm{Cos}(\omega_c t)\mathrm{Sen}(\omega_c t)$$

Equation:

$$y'_1(t) = \left[\frac{x_1(t)}{2} - \frac{x_1(t)\text{Cos }(2\ \omega_c t)}{2} + \frac{x_2(t)\text{Sen }(2\ \omega_c t)}{2}\right]$$

Al pasar por un filtro pasabajos esta señal se eliminan las componentes de alta frecuencia:

Equation:

$$y_1(t) = \frac{x_1(t)}{2}$$

De manera similar ocurre con la otra señal:

Equation:

$$y'_2(t) = [x_1(t)\text{Sen}(\omega_c t) + x_2(t)\text{Cos}(\omega_c t)]\text{Cos}(\omega_c t)$$

Equation:

$$y'_2(t) = x_1(t)\text{Sen}(\omega_c t)\text{Cos}(\omega_c t) + x_2(t)\text{Cos}^2(\omega_c t)$$

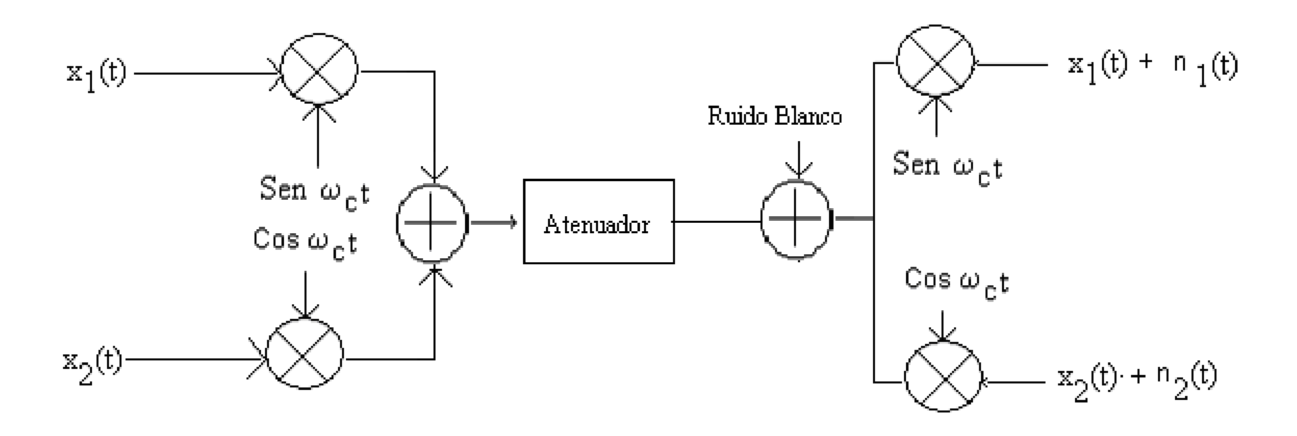
Equation:

$$y'_2(t) = \left[\frac{x_2(t)}{2} + \frac{x_2(t)\text{Cos }(2\ \omega_c t)}{2} + \frac{x_1(t)\text{Sen }(2\ \omega_c t)}{2}\right]$$

Equation:

$$y_2(t) = \frac{x_2(t)}{2}$$

En la figura 3 se muestra un **sistema** completo (Transmisor, Canal y Receptor) que usa este principio.



Sistema DSB en Cuadratura

Autoevaluación

Exercise:

Problem:

¿Qué se obtendrá si una señal modulada en cuadratura como la de la ecuación 2 se demodula con una señal $\sin(\omega_c t + 45^\circ)$?

Solution:

$\sin(\omega_c t + 45^\circ)$ se puede escribir como $\sin(45^\circ)\cos(\omega_c t) + \cos(45^\circ)\sin(\omega_c t)$, es decir, se está demodulando la señal con la suma de un seno más un coseno, por lo que se obtendrá a la salida una combinación de ambos mensajes originales.

Exercise:

Problem:

¿Qué se obtendrá si una señal modulada en cuadratura como la de la ecuación 2 se demodula con una señal $\cos(\omega_c t + 90^\circ)$?

Solution:

$\cos(\omega_c t + 90^\circ)$ es igual a $-\sin(\omega_c t)$, por lo que se obtendrá el mensaje $x_1(t)$ con signo negativo. Para señales de audio, el efecto del signo negativo no se hace notar.

Simuladores

ESTE VINCULO contiene una carpeta con un programa realizado en **MATLAB** que simula un sistema de modulación DSB en Cuadratura. La carpeta incluye el .m y todos los archivos necesarios para su funcionamiento, si se elimina o renombra alguno de estos archivos, el programa podría no funcionar correctamente. La figura 4 contiene un video explicativo acerca del uso del programa.

Modulador en Cuadratura en MATLAB

[missing_resource: http://www.youtube.com/v/OCbUGpi3GHI?fs=1&hl=es_ES]

Video explicativo de la utilización del programa realizado en MATLAB

Puede obtenerse también un programa realizado en **LabVIEW** acerca del mismo tema por medio de ESTE VINCULO. La carpeta incluye el .vi y todos los archivos necesarios para su funcionamiento. Igualmente, si se elimina o renombra alguno de estos archivos, el programa podría no funcionar correctamente. La figura 5 contiene un video explicativo acerca del uso del programa

Modulador en Cuadratura LabVIEW

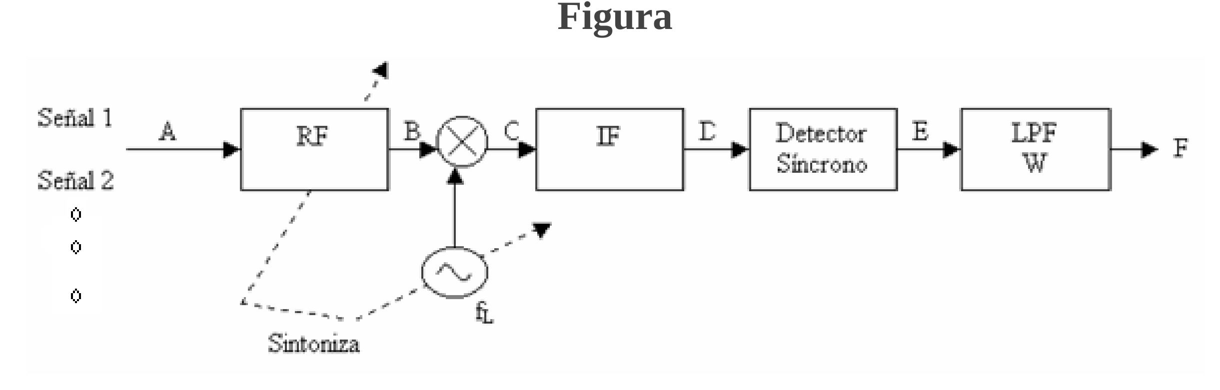
[missing_resource: http://www.youtube.com/v/7D_JpFDOhKE?fs=1&hl=es_ES]

Video explicativo de la utilización del programa realizado en LabVIEW

Receptor Superheterodino para detectar emisoras AM
Receptor superheterodino y detección coherente. Se incluye un programa en MATLAB y otro en LabVIEW, cada uno simula un sistema que modula varias señales de voz y las recupera por medio de un receptor superheterodino.

Cuando se transmiten **señales** usando un mismo medio, el receptor debe seleccionar la banda específica que corresponde a la señal que quiere rescatarse del canal. Un ejemplo de esta situación está en la radiodifusión comercial de señales AM. Los radios comerciales (tanto AM como FM) utilizan la estructura de un Receptor Superheterodino, un **sistema** cuyo esquema se observa en la figura 1:

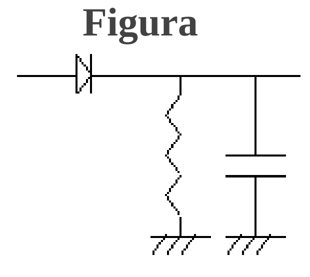
Figura



Esquema de un receptor superheterodino

En algunos de ellos se sustituyen los dos últimos bloques de detección coherente por bloques de detección no coherente, dichos bloques corresponden con un detector de envolvente como el mostrado en la figura 2 y por en eliminador de DC

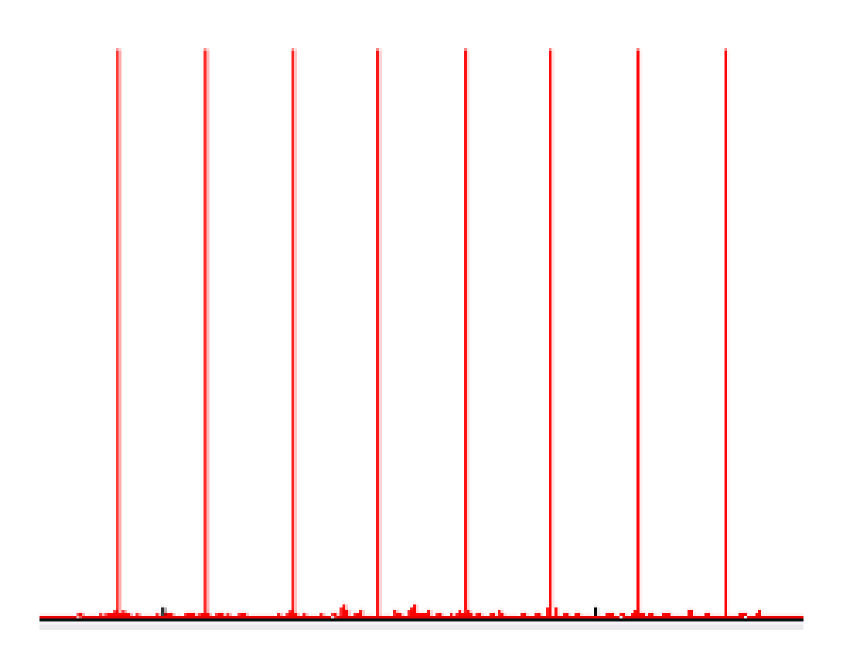
Figura



Detector de envolvente

A la entrada del receptor superheterodino se consigue la parte del espectro electromagnético conformado por la suma de todos los canales posibles de escuchar:

Figura



Espectro de frecuencias que incluye todas las emisoras

El filtro RF es un BPF de un orden no muy alto y con un ancho de banda no tan estrecho como para dejar pasar un solo canal, sino que deja pasar más señal de la deseada. El mezclador (multiplicador por un coseno) efectúa el producto de la señal que está a la salida del filtro RF (punto B en la figura 1) y una sinusoide proveniente del oscilador local de frecuencia $\mathbf{f_L}$; si la entrada fuese una señal centrada en $\mathbf{f_{IN}}$, la frecuencia del oscilador local se elige como $\mathbf{f_L = f_{IN} + f_{IF}}$, en un radio comercial, la frecuencia central del filtro RF y el oscilador local están lógica o mecánicamente conectados para que esto se cumpla. $\mathbf{f_{IF}}$ es un valor de frecuencia conocido como frecuencia

intermedia menor a la mínima frecuencia entre los posibles canales a escuchar que coincide con la frecuencia central del filtro IF. El tener una baja frecuencia de operación, permite a este filtro ser muy selectivo y así, poder tomar un solo canal.

Al mezclar $\mathbf{f_{IN}}$ con $\mathbf{f_L = f_{IN} + f_{IF}}$ aparecerán dos valores de frecuencia, una frecuencia suma centrada en $\mathbf{2f_{IN} + f_{IF}}$ que no pasará por el filtro IF, y una frecuencia resta centrada en $\mathbf{f_{IF}}$ que claramente si pasará. Lo que se ha conseguido con todo esto es “mover” el canal que estaba centrado en $\mathbf{f_{IN}}$ a una frecuencia más baja llamada $\mathbf{f_{IF}}$ para lograr hacer un mejor procesamiento y filtraje a la señal recuperada, además de ser el mismo para cada canal o emisora que se desee escuchar.

Por último viene un demodulador para extraer de la señal, el cual puede ser un detector coherente o no coherente. El detector coherente está formado por un oscilador de frecuencia $\mathbf{f_{IF}}$ que al multiplicarse por la señal la lleva a banda base, el mismo está seguido por un filtro pasabajo encargado de evitar la frecuencia imagen para lo cual debe tener un ancho de banda menor a $\mathbf{2f_{IF}}$, lo normal es que el ancho de banda coincida con el del mensaje.

Autoevaluación

Exercise:

Problem:

¿Qué sucede si se escoge un valor de frecuencia intermedia menor al ancho de banda del canal?

Solution:

Si esto ocurre, al trasladarse el canal a esta frecuencia, una porción del mismo quedará en el eje negativo de las frecuencias y una porción de diferente tamaño en el eje positivo, lo cual dañaría la señal.

Exercise:

Problem:

¿Puede escogerse un valor de frecuencia intermedia igual a la frecuencia en la que originalmente se centra una de las emisoras?

Solution:

Podría escogerse un valor de frecuencia intermedia ubicado en cualquier punto del espectro mostrado en la figura 3 sin problema alguno, aunque lo mismo no suele hacerse cuando lo que se busca es bajar la frecuencia para facilitar el procesamiento de la señal.

Simuladores

ESTE VINCULO contiene una carpeta con un programa realizado en **MATLAB** que simula un Receptor Superheterodino para detectar emisoras AM. La carpeta incluye el .m y todos los archivos necesarios para su funcionamiento, si se elimina o renombra alguno de estos archivos, el programa podría no funcionar correctamente. La figura 4 contiene un video explicativo acerca del uso del programa.

Receptor Superheterodino MATLAB

[missing_resource: http://www.youtube.com/v/xIZQnQrLsQY?fs=1&hl=es_ES]

Video explicativo de la utilización del programa realizado en MATLAB

Puede obtenerse también un programa realizado en **LabVIEW** acerca del mismo tema por medio de ESTE VINCULO. La carpeta incluye el .vi y todos los archivos necesarios para su funcionamiento. Igualmente, si se elimina o renombra alguno de estos archivos, el programa podría no funcionar correctamente. La figura 5 contiene un video explicativo acerca del uso del programa.

Receptor Superheterodino LabVIEW

[missing_resource: http://www.youtube.com/v/v12Yd0Bw0wM?fs=1&hl=es_ES]

Video explicativo de la utilización del programa realizado en LabVIEW

Filtraje Óptimo para detección de eventos inmersos en ruido
Señales completamente inmersas en ruido son rescatadas por medio de sistemas conocidos como Filtros Óptimos. Se incluye un programa en MATLAB y otro en LabVIEW, cada uno aplica el filtraje optimo para recuperar información en diversos tipos de señales, entre ellas una señal electrocardiográfica.

Se tiene una forma de onda básica p(t) existente entre 0 y un valor D definido como la duración de dicha forma de onda, la misma se repite en el tiempo y se contamina con ruido en el canal de transmisión, la expresión para cada repetición de p(t) viene dada por:

Equation:

La **señal** resultante se definirá como una sumatoria de ruido más expresiones semejantes a la ecuación 1 pero con diferentes valores de t_0, adicionalmente algunas de las repeticiones podrían estar multiplicadas por alguna constante. Esta señal se puede filtrar con un **sistema** cuya respuesta impulsiva toma la siguiente forma:

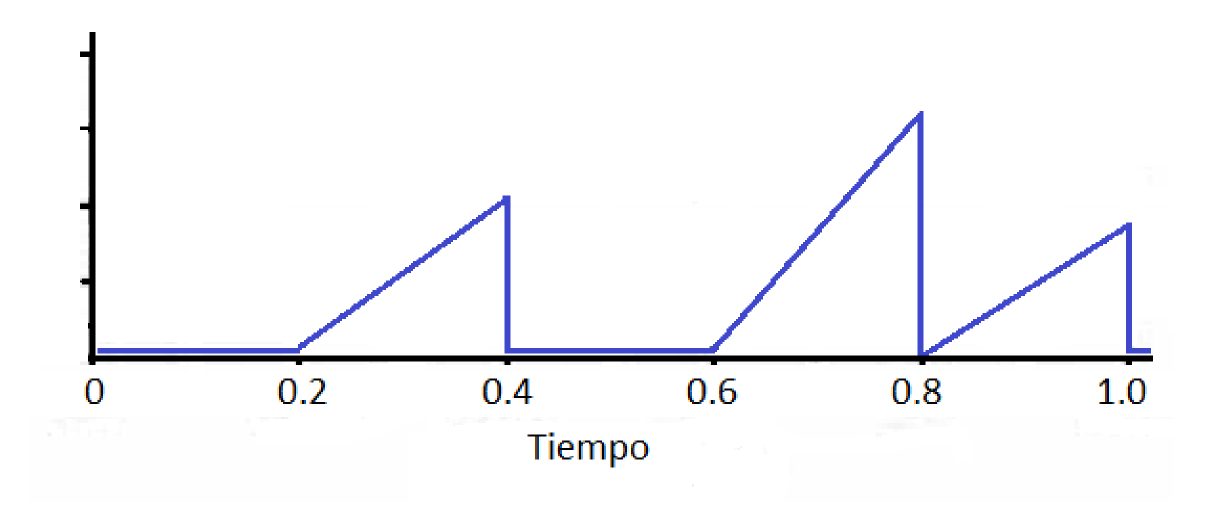
Equation:

Nótese que la variable t tiene signo negativo, esto se traduce en que dicha respuesta impulsiva está invertida con respecto al eje de las ordenadas.

Es necesario recordar que una forma de encontrar la señal de salida de un sistema en el dominio del tiempo es **convolucionando** la señal de entrada en el dominio del tiempo con la respuesta impulsiva del sistema. De esta forma se obtiene a la salida una señal con valores máximos situados en los puntos de ocurrencia de cada repetición.

Supóngase como señal de entrada la presente en la figura 1:

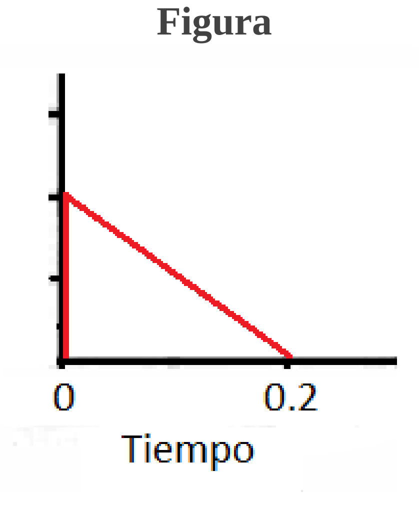
Figura



Señal de entrada

En vista de las formas de onda presentes, para las cuales el valor D es de 0.2, el filtro óptimo tendrá una respuesta impulsiva como la siguiente:

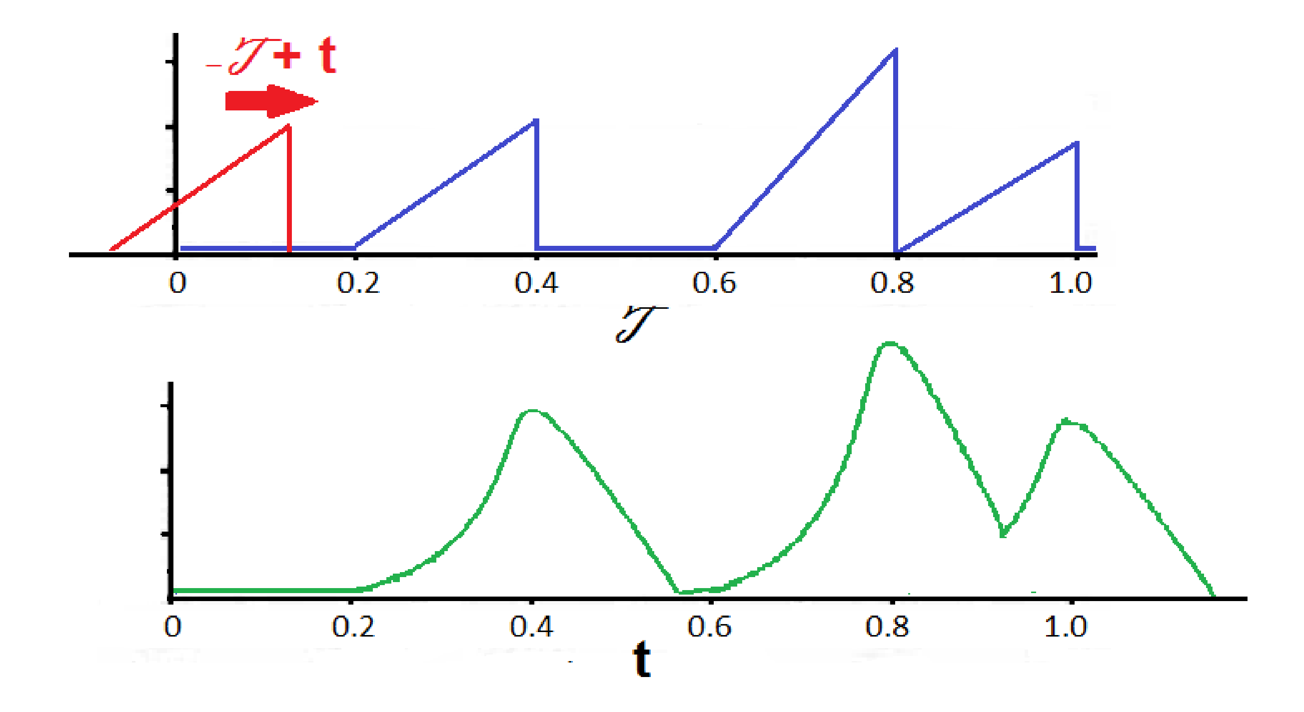
Figura



Respuesta impulsiva
del filtro óptimo

Al realizar la convolución entre la señal y el filtro, se obtendrá como resultado una señal parecida a la mostrada en verde en la figura 3:

Figura



Proceso de convolución y señal resultante aproximada.

Esto se ha aplicado en diversas áreas, desde detección de señales digitales, ubicación de complejos QRS en un electrocardiograma, detección de capas geológicas para descubrir posibles yacimientos petroleros, etc.

Autoevaluación

Exercise:

Problem:

¿Cómo debe ser la respuesta impulsiva del filtro óptimo para una señal digital binaria formada por pulsos rectangulares de duración Tbit?

Solution:

La respuesta impulsiva será igual al pulso rectangular y tendrá una duración de Tbit, esto se debe a que si se invierte un pulso rectangular

con respecto al eje vertical la forma de onda no cambiará (aplica para cualquier forma de onda horizontalmente simétrica).

Exercise:

Problem:

¿Cómo varía la señal de salida si el filtro óptimo no toma su valor inicial en t=0?

Solution:

Si el valor inicial del filtro óptimo se sitúa en t=0.3, por ejemplo, la única diferencia en la señal de salida será que la misma tendrá un desplazamiento hacia la izquierda de 0.3. Si esto ocurre en una aplicación de la vida real, es importante conocer el mencionado tiempo y tomar en cuenta el desplazamiento.

Simuladores

ESTE VINCULO contiene una carpeta con un programa realizado en **MATLAB** que aplica el Filtraje Óptimo a señales contaminadas con ruido. La carpeta incluye el .m y todos los archivos necesarios para su funcionamiento, si se elimina o renombra alguno de estos archivos, el programa podría no funcionar correctamente. La figura 4 contiene un video explicativo acerca del uso del programa.

Filtraje Óptimo en MATLAB

[missing_resource: http://www.youtube.com/v/_DlQJvRkyVI?fs=1&hl=es_ES]

Video explicativo de la utilización del programa realizado en MATLAB

Puede obtenerse también un programa realizado en **LabVIEW** acerca del mismo tema por medio de ESTE VINCULO. La carpeta incluye el .vi y todos los archivos necesarios para su funcionamiento. Igualmente, si se

elimina o renombra alguno de estos archivos, el programa podría no funcionar correctamente. La figura 5 contiene un video explicativo acerca del uso del programa.

Filtraje Óptimo en LabVIEW

[missing_resource: http://www.youtube.com/v/2if80OBWN2Q?fs=1&hl=es_ES]

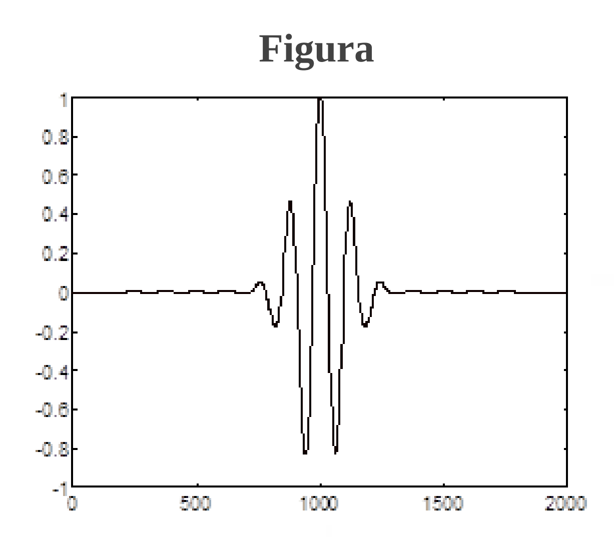
Video explicativo de la utilización del programa realizado en LabVIEW

La Transformada Ondícula y sus aplicaciónes
En este módulo se explica el concepto y el proceso de la transformada ondícula, además de algunas aplicaciones. Se incluye un programa hecho en el software MATLAB y un programa hecho en el software LabVIEW, cada uno muestra las aplicaciones de esta transformada. Se incluyen también videos explicativos para el uso de los programas

La **Transformada de Fourier** puede ser vista como la proyección de la **señal** x(t) sobre las **bases** ortogonales exponenciales (senos y cosenos). También puede verse como el análisis de la señal en bandas de frecuencias uniformes:

Equation:

Figura



Ondícula de Morlet

La Transformada Continua de Ondícula (*Continuos Wavelet Transform*) se define como:

Equation:

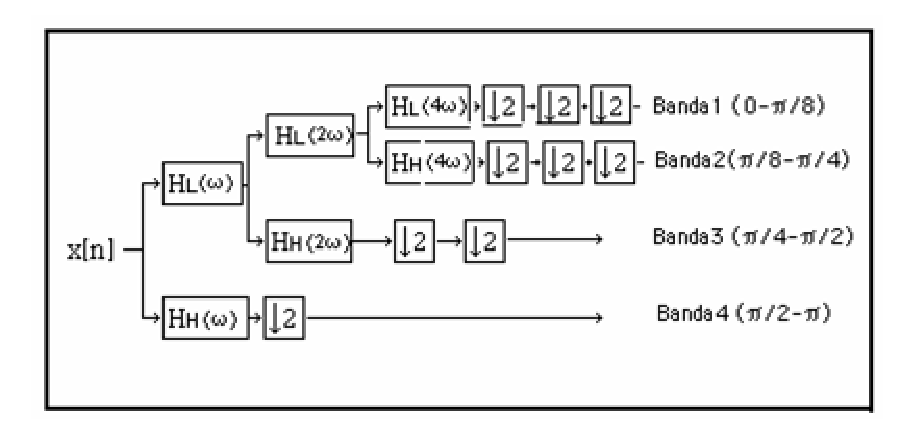
Se observa que se hace la proyección de la señal x(t) sobre versiones escaladas y desplazadas de una ondícula madre llamada w(t). Basado en el ejemplo anterior se puede inferir que la transformada ondícula parece más apropiada que la de Fourier para señales abruptas, cambiantes, no repetitivas, en fin casi todas las señales del mundo real.

Si en vez de pensar en una transformada continua se plantea una discreta limitando los valores de a y b a potencias de 2, aparece la Transformada Ondícula Discreta o DWT la cual, en el dominio de la frecuencia se plantea como:

Equation:

Una técnica utilizada para realizar la Transformada Ondícula Discreta es la descomposición en bandas no uniformes (descomposición en octavas), utilizando filtros pasabajos y pasaaltos específicos que dividen toda la gama de frecuencias en bandas no uniformes. Por ejemplo, si se usa una descomposición de **profundidad 3** el **sistema** luciría como muestra la figura 2. Se incluyen diezmadores (el ‘2’ en el recuadro) que eliminan una de cada dos muestras, esto es para no aumentar el número de puntos a la salida.

Figura

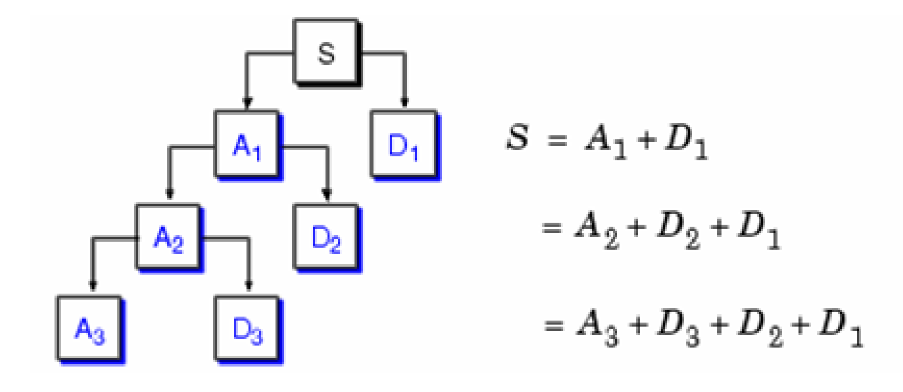


Sistema de la DWT

Para ilustrar la labor de los diezmadores, supóngase que la señal original x(n) tiene 1000 puntos, la señal en la banda 4 es diezmada una vez, por lo que su longitud será de 500 puntos; la señal en la banda 3 se diezma 2 veces por lo que su longitud será de 250 puntos ($1000/2^2$), y las señales en las bandas 1 y 2 se diezman 3 veces, quedando con una longitud de 125 ($1000/2^3$).

Utilizando este esquema no uniforme se puede reconstruir la señal invirtiendo el proceso de filtraje, es decir, utilizando filtros de reconstrucción apropiados, se filtran las señales de salida de cada rama pasaaltos, y la salida de la última rama pasabajos, para obtener la señal original. Esquemáticamente para hacer la descomposición en ondículas se usa un árbol como el siguiente:

Figura

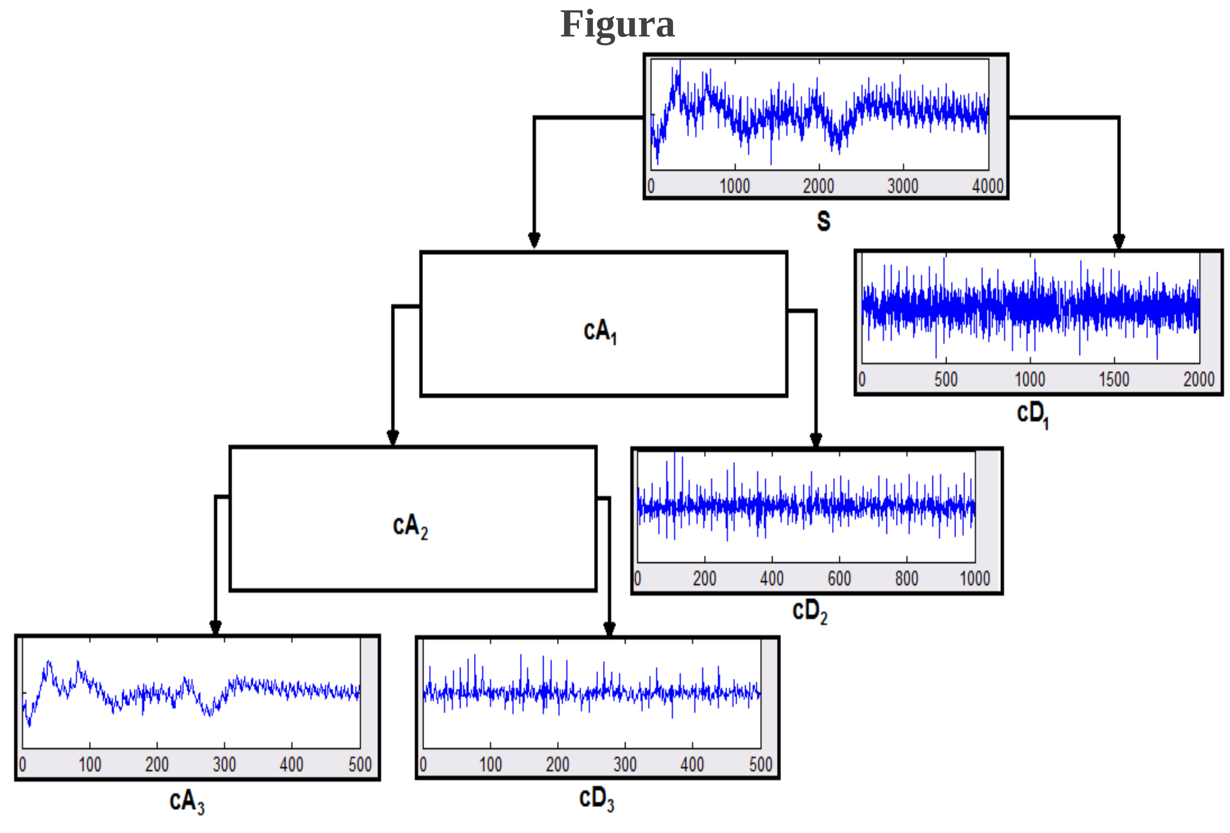


Esquema de la descomposición en ondículas

La señal S se pasa por filtros pasaaltos y pasabajos; las salidas de los pasaaltos reciben el nombre de detalles (cD_1, cD_2, etc...), a las de los pasabajos se les llama cA_k.

Por ejemplo. Una señal S se descompone usando un árbol como el anterior y las salidas serían:

Figura



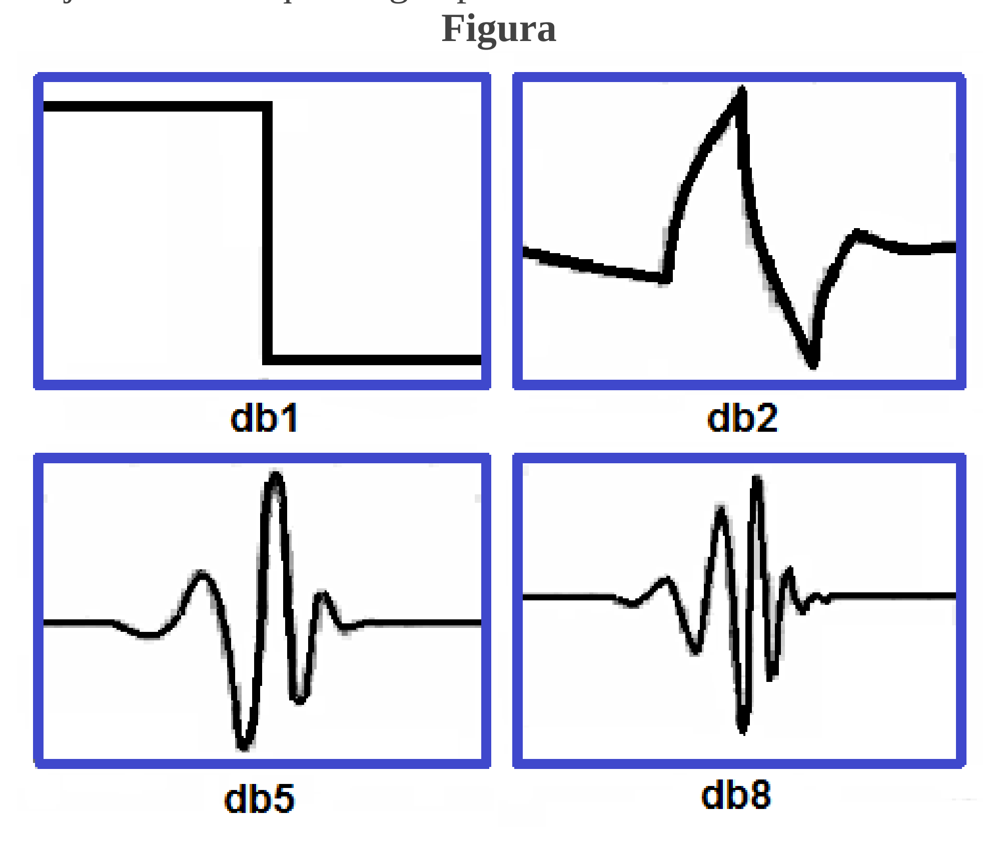
Señal descompuesta

Este tipo de análisis permite hacer algún procesamiento en la salida de los filtros de descomposición (por ejemplo, eliminar un detalle que no aporte información relevante o con mucho ruido ruido, o simplemente analizarlo para identificar un evento determinado), para una vez invertido el proceso

simplificar el análisis. De esta misma forma, se puede realizar **compresión** de datos y supresión de ruido.

Las ondículas madres más usadas son las Daubechies y se identifican como: 'db1', ‘db2’...’db10’... 'db45', en la figura 5 se muestran algunas de ellas. Obsérvese que unas son más abruptas que otras, por lo tanto se adaptarán mejor a señales que tengan parecido con ellas.

Figura



Ondículas madre tipo Daubechies

La transformada Ondícula para reducción de ruido (denoising)

Este método es mejor que simplemente filtrar la señal contaminada con un filtro pasabajos ya que esto puede eliminar los cambios abruptos que aparecen en la señal y que son importantes en ella; el filtraje pasabajos también es incapaz de eliminar ruido que está en la misma banda de

frecuencia de la señal; en la transformada ondícula la separación entre ruido y señal no es por frecuencia.

Se tiene una señal x contaminada con ruido blanco gaussiano de media cero. Esto produce una señal que será llamada Y. La idea es recobrar x lo mejor posible. El procedimiento sería el siguiente: Se descompone la señal Y usando una determinada ondícula madre y un determinado nivel de descomposición. Para cada uno de los Detalles resultantes de la descomposición se le aplica una eliminación de coeficientes por umbral; los detalles así procesados se utilizan para reconstruir la señal de nuevo que debiera estar más limpia. Hay dos formas de aplicar la eliminación por umbral una Suave (soft) y una Dura (hard). Dado un umbral T, la Soft funciona de la siguiente forma a la salida del proceso se tendrá una señal Z tal que:
Equation:

En cambio la Hard produce la señal Z tal que:
Equation:

Para elegir el nivel de umbral más apropiado existen diferentes métodos. El software MATLAB incluye algunos de ellos: ‘rigrsure’, ‘heursure’, ‘sqtwolog’, ’minimaxi’.

Autoevaluación

Exercise:

Problem:

Aplicando la transformada ondícula de profundad 5 a una señal, la componente cA_5 tiene una longitud de 500 puntos, ¿Qué longitud tiene la componente cD_1?

Solution:

La longitud de la componente resulta cA_5 de dividir la longitud total entre 2^5, despejando se obtiene una longitud total de 500x32=16000 puntos, el primer detalle tiene la mitad de la longitud total, es decir, 8000 puntos.

Exercise:

Problem:

Aplicando la transformada ondícula de profundidad 4 a una señal muestreada a 8000Hz, ¿en qué componente se puede apreciar mejor la presencia de un tono de 440Hz?

Solution:

Si la señal ha sido muestreada a 8000Hz la gama de frecuencias irá desde 0 hasta la mitad de la frecuencia de muestreo: 4000Hz. De la descomposición de profundidad 4 se obtendrán las siguientes bandas: $cD_1 \rightarrow$ 2000-4000Hz, $cD_2 \rightarrow$ 1000-2000Hz, $cD_3 \rightarrow$ 500-1000Hz, $cD_4 \rightarrow$ 250-500Hz, $cA_4 \rightarrow$ 0-250Hz. El tono de 440Hz será mejor apreciado en el detalle cD_4.

Exercise:

Problem:

¿Cuántas componentes será necesario conservar al comprimir un señal electrocardiográfica, si la única información requerida es el instante de ocurrencia de cada ciclo?

Solution:

Sólo bastaría con conservar una componente, específicamente la que contiene la frecuencia de repetición de cada ciclo.

Exercise:

Problem:

¿Cuál de las 4 ondículas madre en la figura 5 será más recomendable al aplicar la transformada ondícula a una señal de la cual desea observarse más detalladamente la componente con mayor frecuencia?

Solution:

La ondícula madre cuadrada (db1) es la que presenta el cambio más abrupto. Los cambios abruptos se asocian con altas frecuencias, lo que hace a esta ondícula madre la más recomendable para detallar la componente de mayor frecuencia (cD_1).

Simuladores

ESTE VINCULO contiene una carpeta con un programa realizado en **MATLAB** que muestra varias aplicaciones de la Transformada Ondícula. La carpeta incluye el .m y todos los archivos necesarios para su funcionamiento, si se elimina o renombra alguno de estos archivos, el programa podría no funcionar correctamente. La figura 6 contiene un video explicativo acerca del uso del programa.

Transformada Ondícula MATLAB

[missing_resource: http://www.youtube.com/v/BjNXtMmg19w?fs=1&hl=es_ES]

Video explicativo de la utilización del programa realizado en MATLAB

Puede obtenerse también un programa realizado en **LabVIEW** acerca del mismo tema por medio de ESTE VINCULO. La carpeta incluye el .vi y todos los archivos necesarios para su funcionamiento. Igualmente, si se

elimina o renombra alguno de estos archivos, el programa podría no funcionar correctamente. La figura 7 contiene un video explicativo acerca del uso del programa.

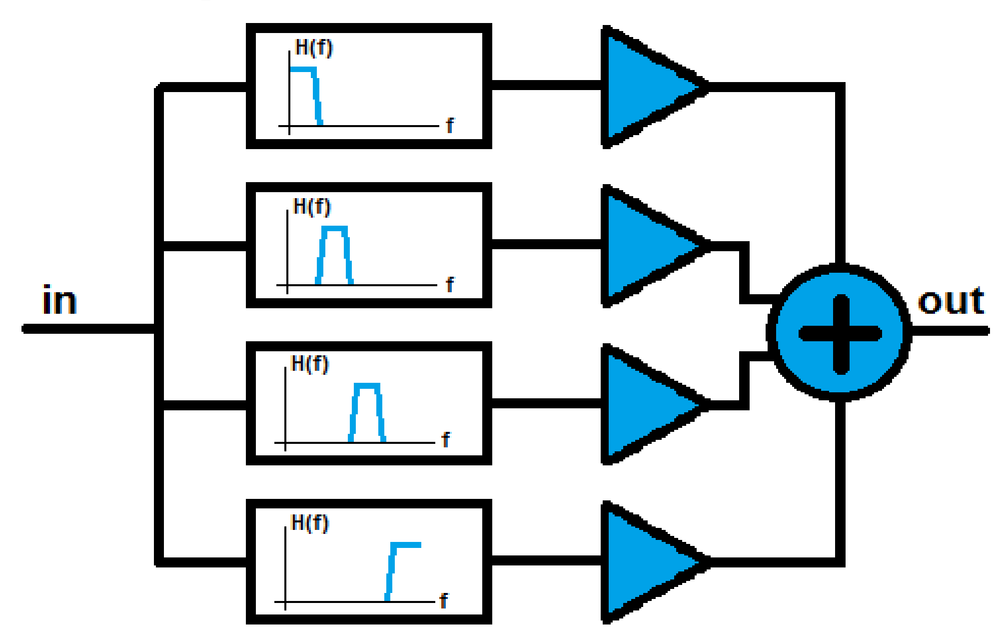
Transformada Ondícula LabVIEW

[missing_resource: http://www.youtube.com/v/oD7DncU7FpY?fs=1&hl=es_ES]

Video explicativo de la utilización del programa realizado en LabVIEW

Ecualizador y Sintetizador Musical
Incluye programas en MATLAB y LabVIEW que simulan un ecualizador gráfico y sintetizador musical. Fórmulas para calcular frecuencias de las notas musicales por octavas

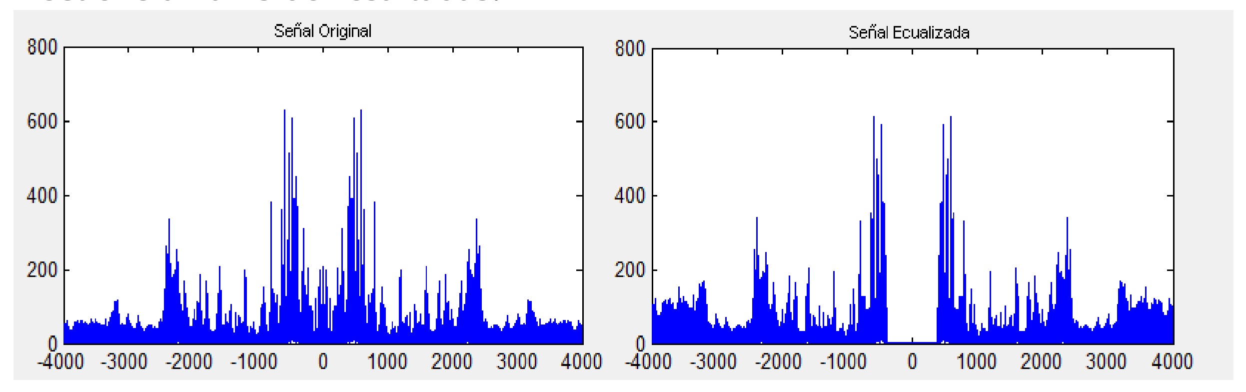
Un ecualizador es un **sistema** electrónico utilizado para modificar las características de alguna **señal** auditiva, resaltando o atenuando la intensidad de cada una de las bandas de frecuencia que componen la señal; un ecualizador puede ser utilizado para comprimir señales auditivas atenuando completamente las bandas de frecuencia muy con poco aporte. Este dispositivo está constituido por una cantidad determinada de filtros colocados en paralelo a los cuales entra la señal original, la salida del ecualizador se constituye por la suma de la señal de salida de cada uno de los filtros. En la figura 1 se muestra un ecualizador de 4 bandas uniformes, cada filtro está seguido de un amplificador/atenuador para luego sumarse con la señal proveniente de los demás filtros:



Ecualizador de 4 bandas

En la figura 2 se muestra una señal muestreada a 8KHz pasada por un ecualizador de 10 bandas uniformes, nótese como la banda de menor

frecuencia ha sido totalmente atenuada y las dos bandas de mayor frecuencia han sido resaltadas:



Señal ecualizada

Sintetizador Musical

Un sintetizador es un dispositivo electrónico utilizado para generar música de forma artificial, puede definirse como un instrumento musical electrónico. Algunos de ellos son capaces de generar sonidos exactamente iguales a los de ciertos instrumentos musicales; una modalidad sencilla de estos dispositivos es capaz de generar tonos auditivos logrados por medio de la reproducción de ondas sinusoidales de cierta duración y frecuencia, las notas musicales reproducidas vienen dadas por la frecuencia de cada sinusoide. Los valores de frecuencia para las octavas de cada nota musical están presentes en la siguiente tabla:

Notas	**Frecuencia por octavas (Hz)**				
La	55.00	110.00	220.00	440.00	880.00
La #	58.27	116.54	233.08	466.16	932.32

Si	61.74	123.48	246.96	493.92	987.84
Do	65.41	130.82	261.64	523.28	1046.56
Do #	69.30	138.60	277.20	554.40	1108.80
Re	73.42	146.84	293.68	587.36	1174.72
Re #	77.78	155.56	311.12	622.24	1244.48
Mi	82.41	164.82	329.64	659.28	1318.56
F a	87.31	174.62	349.24	698.48	1396.96
F a#	92.50	185.00	370.00	740.00	1480.00
Sol	98.00	196.00	392.00	784.00	1568.00
Sol#	103.83	207.66	415.32	830.64	1661.28

Valores de frecuencia para las notas musicales. El sintetizador realizado en MATLAB es capaz de reproducir los valores entre 261.64 y 554.40Hz y el sintetizador realizado en LabVIEW reproduce los valores entre 261.64 y 987.84Hz. Los sintetizadores se pueden descargar en la sección "Simuladores"

Cada valor de frecuencia se obtiene al multiplicar el valor anterior por $2^{1/12}$:
Equation:

Aplicando la ecuación 1 al valor para la primera octava de "Fa#" se obtiene el valor de frecuencia de "Sol":
Equation:

Generalizando la ecuación 1 se obtiene:
Equation:

Si se aplica la ecuación 3 al valor para la primera octava de “La” con N=7 se obtiene el valor de frecuencia de “Mi”:
Equation:

Como puede observarse en la tabla 1, si se duplica el valor de frecuencia de una nota musical, se obtendrá el valor de frecuencia de la misma nota pero para la siguiente octava, esto se demuestra por medio de la ecuación 3 con N=12:
Equation:

Para lograr generar y escuchar un tono de duración 0.5 segundos y con frecuencia de 440Hz en el software **MATLAB** basta con las siguientes instrucciones:

```
T=1/8000;
t=(0:T:0.5-T);
tono=sin(2*pi*440*t);
sound(tono,1/T);
```

Instrucciones para la reproducción de un tono de 440Hz de

frecuencia en MATLAB

Y para lograr generar en **LabVIEW** este mismo tono basta con el siguiente diagrama:

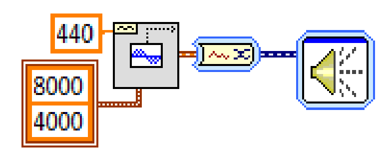


Diagrama de bloques para la reproducción de un tono de 440Hz de frecuencia en LabVIEW

Autoevaluación

Exercise:

Problem:

Si una señal musical es procesada con un ecualizador, ¿qué frecuencias deben acentuarse si se desea resaltar el sonido de una guitarra eléctrica? ¿y cuáles deben atenuarse si desea suprimir la voz del cantante?

Solution:

Instrumentos como la guitarra eléctrica tienen un sonido agudo, por lo cual son asociados con frecuencias altas. Para suprimir la voz del cantante basta con suprimir las frecuencias por debajo de 2KHz; si el ecualizador lo permite, pueden conservarse las frecuencias por debajo de 150Hz para evitar suprimir sonidos de instrumentos graves.

Exercise:

Problem:

¿Es posible eliminar el nivel DC de una señal con un ecualizador?

Solution:

Sí es posible eliminar el nivel DC atenuando al máximo la primera banda, el problema es que también se estarían atenuando las componentes AC de esa gama de frecuencias.

Exercise:

Problem:

Se tiene una señal cuadrada periódica, la misma se hace pasar por un ecualizador cuyos filtros son muy estrechos y están centrados alrededor de cada una de las armónicas impares. ¿Podría conseguirse a la salida una sinusoide pura?

Solution:

Sí. Atenuándose al máximo todas las bandas excepto una se tendrá a la salida sólo una delta, lo que corresponde con el comportamiento en frecuencia de una sinusoide.

Simuladores

ESTE VINCULO contiene una carpeta con un programa realizado en **MATLAB** que simula un Ecualizador Gráfico y un sintetizador Musical. La carpeta incluye el .m y todos los archivos necesarios para su funcionamiento, si se elimina o renombra alguno de estos archivos, el

programa podría no funcionar correctamente. La figura 5 contiene un video explicativo acerca del uso del programa.

Ecualizador/Sintetizador MATLAB

[missing_resource: http://www.youtube.com/v/WDdXFHnvDwo?fs=1&hl=es_ES]

Video explicativo de la utilización del programa realizado en MATLAB

Puede obtenerse también un programa realizado en **LabVIEW** acerca del mismo tema por medio de ESTE VINCULO. La carpeta incluye el .vi y todos los archivos necesarios para su funcionamiento. Igualmente, si se elimina o renombra alguno de estos archivos, el programa podría no funcionar correctamente. La figura 6 contiene un video explicativo acerca del uso del programa.

Ecualizador/Sintetizador LabVIEW

[missing_resource: http://www.youtube.com/v/MvaQH2BteCQ?fs=1&hl=es_ES]

Video explicativo de la utilización del programa realizado en LabVIEW

Ortogonalización Gram-Schmidt
Método Gram-Schmidt para cálculo de bases ortogonales por medio de los símbolos de una señal. Se explica también el concepto de constelación en comunicaciones. Se incluyen dos programas en LabVIEW acerca de este tema, uno de los dos hecho por medio de MATLAB Script

El siguiente módulo está hecho en base al módulo ORTOGONALIZACIÓN GRAM-SCHMIDT Y TEORÍA BÁSICA DE LAS CONSTELACIONES, realizado por *Venuska González y Mariangela Mezoa.*

En matemáticas, el concepto de **Ortogonalidad** está referido al de **Perpendicularidad**. Se dice que dos vectores $\vec{x}$ y $\vec{y}$ pertenecientes a cierto espacio vectorial (V) son ortogonales si se cumple que el producto escalar entre ellos es igual a **cero,** es decir:
Equation:

$$\vec{x} \cdot \vec{y} = 0$$

A partir de un conjunto de vectores linealmente independientes se puede construir un nuevo conjunto de vectores **ortonormales** (Que cumplan con las condiciones de ortogonalidad y norma vectorial). Esto se conoce como el método de **OrtogonalizaciónGram-Schmidt (G-S).** Pero, ¿cómo aplicar este concepto para un sistema de comunicación digital?

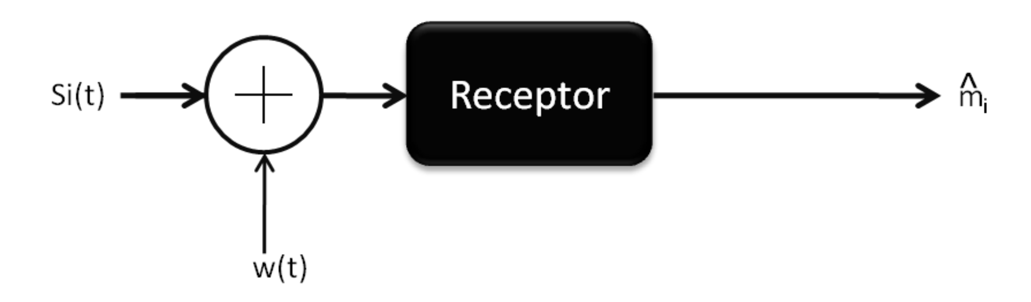
Ortogonalización Gram-Schmidt

Supóngase una señal $S_i(t)$ que representa a un símbolo m_i. Se estima que esta señal pase por el receptor que está encargado de obtener cada símbolo de la misma. Sin embargo, es evidente que al pasar por el canal, la señal se contaminará debido a la existencia de ruido en el sistema. En una condición ideal, el resultado sería el siguiente:



Sistema ideal de recepción (sin ruido). Cada símbolo mi de la señal se recibe sin interferencia.

Al introducir ruido blanco gaussiano en el **sistema**, quedaría como sigue:



Sistema de recepción con introducción de ruido

La segunda situación ocasiona que a la salida del receptor no se obtenga precisamente el símbolo m_i, sino que se obtenga un **estimado** del símbolo original.

Es en este punto en donde entra el concepto de ortogonalización G-S: La señal $S_i(t)$ puede expresarse en función de un conjunto finito de bases (o vectores) ortonormales (**U**), de forma tal que cada forma de onda estaría relacionada con un coeficiente que será denominado **s**. Matemáticamente se tiene que:

Equation:

$$S_i(t) = \sum_{j=1}^{n} s_{\mathrm{ij}}.U_j(t)$$

Es decir, a cada símbolo m_i se le asocia una forma de onda s. desarrollando la fórmula anterior, **para todos los símbolos posibles**, se obtiene un sistema de ecuaciones como sigue:

Equation:

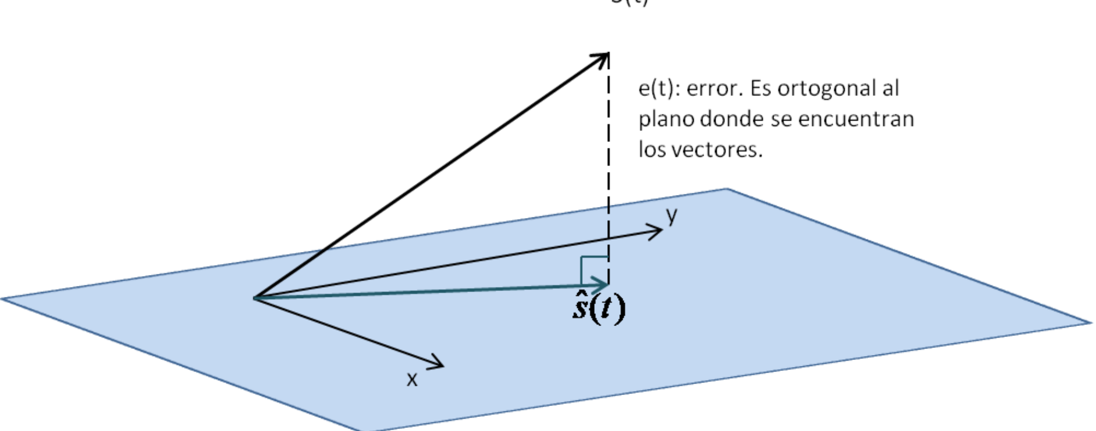
$$\begin{gathered} S_1(t) = s_{11}.U_1(t) + s_{12}.U_2(t) + s_{13}.U_3(t) + ... + s_{1\mathrm{n}}.U_n(t) \\ S_2(t) = s_{21}.U_1(t) + s_{22}.U_2(t) + s_{23}.U_3(t) + ... + s_{2\mathrm{n}}.U_n(t) \\ S_3(t) = s_{31}.U_1(t) + s_{32}.U_2(t) + s_{33}.U_3(t) + ... + s_{3\mathrm{n}}.U_n(t) \\ \vdots \\ S_m(t) = s_{m1}.U_1(t) + s_{m2}.U_2(t) + s_{m3}.U_3(t) + ... + s_{\mathrm{mn}}.U_n(t) \end{gathered}$$

El objetivo cuando se tiene un sistema como el mostrado en la figura 2 es el de obtener el estimado que más se aproxime al valor real. Esto se hace minimizando la energía de la señal de error entre el símbolo original y el estimado:

Equation:

$$\begin{gathered} s_j = \int_0^T S(t).U_j(t)\mathrm{dt} \\ j = 1,2,3,...,N \end{gathered}$$

Visto desde la perspectiva vectorial, el procedimiento será entonces el de obtener una representación de la señal en función de dos vectores en el plano. El estimado del vector original sería entonces la proyección de éste sobre el plano:



Ejemplo aplicado a vectores. ŝ(t) es el estimado de cada forma de onda original S(t) y e(t) sería la introducción de ruido en el sistema.

A continuación se explica paso a paso la metodología para la obtención de las bases necesarias para representar cada símbolo de una determinada **señales de potencia**:

Se tiene un conjunto de **señales de energía** $S_i(t)$ con existencia en un intervalo de tiempo [0, T] que se quieren representar por medio de bases U_j, tal y como se indica en el sistema de ecuaciones 3.

Las bases deben cumplir con el principio de ortonormalidad mencionado al principio:

Equation:

$$\int_0^T U_j(t).U_k(t)\mathrm{dt} = \begin{cases} 1 \rightarrow j = k \\ 0 \rightarrow j \neq k \end{cases}$$

Para comenzar se fija $s_{ij} = 0$ exceptuando el primer valor: s_{11}:

Equation:

$$S_1(t) = s_{11}.U_1(t)$$

Se eleva toda la ecuación al cuadrado y se integra en el intervalo [0,T]:
Equation:

$$\int_0^T [S_1(t)]^2 \mathrm{dt} = \int_0^T s_{11^2}.U_{1^2}(t)\mathrm{dt} = s_{11^2}\int_0^T U_1(t).U_1(t)\mathrm{dt}$$

Por el principio de ortonormalidad, la integral de la derecha es igual a 1, quedando s_{11} sólo en función de S_1(t) por lo que se puede despejar:
Equation:

$$s_{11} = \sqrt{\int_0^t [S_1(t)]^2 \mathrm{dt}}$$

Finalmente:
Equation:

$$U_1(t) = \frac{S_1(t)}{s_{11}} = \frac{S_1(t)}{\sqrt{\int_0^t [S_1(t)]^2 \mathrm{dt}}}$$

Con esto se obtiene la primera base para representar la señal. Para calcular U_2(t), se debe restar a S_2(t) su proyección sobre U_1(t); esto cumpliría con la condición de que la base sea ortogonal.

Ahora se fijará Sij=0 exceptuando los valores de s_{21} y s_{22}:
Equation:

$$S_2(t) = s_{21}.U_1(t) + s_{22}.U_2(t)$$

Reordenando esta ecuación queda:
Equation:

$$s_{22}.U_2(t) = S_2(t) - s_{21}.U_1(t)$$

Multiplicando la ecuación por $U_1(t)$ e integrándola en el intervalo [0,T] queda:
Equation:

$$\int_0^T S_2(t).U_1(t)\mathrm{dt} = \int_0^T s_{21}.U_1(t).U_1(t)\mathrm{dt} + \int_0^T s_{22}.U_2(t).U_1(t)\mathrm{dt}$$

Se aplica el principio de ortonormalidad quedando:
Equation:

$$s_{21} = \int_0^T S_2(t).U_1(t)\mathrm{dt}$$

Al igual que para el paso 1, se eleva toda la ecuación 10 al cuadrado y se integra en el intervalo [0,T], quedando como sigue:
Equation:

$$\int_0^T s_{22^2}U_2.U_2(t) = \int_0^T (S_2(t) - s_{21}U_1(t))^2\mathrm{dt}$$

Usando nuevamente el principio de ortonormalidad, queda s_{22} en función de la señal $S_2(t)$, el coeficiente s_{21} y la base $U_1(t)$:
Equation:

$$s_{22} = \sqrt{\int_0^T \left(S_2(t) - s_{21}U_1(t)\right)^2 \mathrm{dt}}$$

Finalmente, despejando y sustituyendo las ecuaciones 13 y 15 en la ecuación 11:

Equation:

$$\Rightarrow U_2(t) = \frac{\left[s_2(t) - \left(\int_0^T S_2(t).U_1(t)\mathrm{dt}\right)U_1(t)\right]}{\sqrt{\int_0^T \left(S_2(t) - \left(\int_0^T S_2(t).U_1(t)\mathrm{dt}\right)U_1(t)\right)^2 \mathrm{dt}}}$$

Se buscarán cuantas bases sean necesarias hasta el punto en el que U_n=0. Se pudiera resumir este proceso de la siguiente forma:

Equation:

$$U_1(t) = \frac{S_1(t)}{\parallel S_1(t) \parallel}$$

Equation:

$$U_2(t) = \frac{S_2(t) - \langle S_2(t), U_1(t)\rangle.U_1(t)}{\parallel S_2(t) - \langle S_2(t), U_1(t)\rangle.U_1(t) \parallel}$$

Equation:

$$U_n(t) = \frac{S_n(t) - \sum_{m=1}^{n-1}\langle S_n(t), U_m(t)\rangle.U_m(t)}{\parallel S_n(t) - \sum_{m=1}^{n-1}\langle S_n(t), U_m(t)\rangle.U_m(t) \parallel}$$

Donde:

Equation:

$$\parallel X \parallel = \sqrt{E_X} = \sqrt{\int_{-\infty}^{+\infty} X^2(t)\mathrm{dt}}$$

$$\langle x(t),y(t)\rangle = \int x(t)y(t)\mathrm{dt}$$

Es importante resaltar que si el proceso de ortogonalización se inicia con una señal diferente a la señal $S_1(t)$, se obtendría un conjunto distinto de bases ortonormales pero igualmente representativa.

Constelación

Es la representación gráfica de cada señal $S_i(t)$ en función de las bases U_i. Cada punto perteneciente a la constelación corresponde a un símbolo de modulación.

Se considerarán los 'ejes' de la gráfica las bases calculadas a partir de la Ortogonalización, es decir, **Uj**. El procedimiento es el siguiente: se debe representar con un punto a la(s) forma(s) de onda $\mathbf{s_i}$ sobre el eje de la base. Supónganse dos señales, que identifican una determinada codificación o modulación, y que pueden representarse con una sola base de acuerdo a las siguientes ecuaciones:

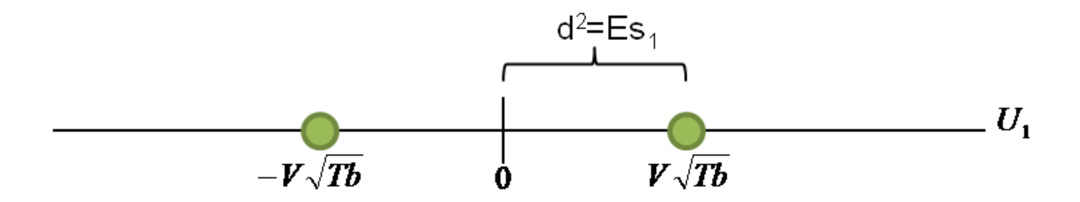
Equation:

$$S_1 = V\sqrt{\mathrm{Tb}}.U_1$$

Equation:

$$S_2 = -V\sqrt{\mathrm{Tb}}.U_1$$

Como sólo se necesita una base para representar estas formas de onda, entonces se tendrá un 'eje' que es U_1:



Ejemplo de constelación.

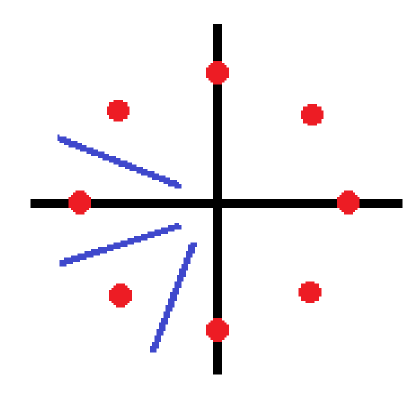
A partir de la constelación se puede obtener un parámetro fundamental que es la **Energía**. Si se eleva al cuadrado la distancia que existe entre el origen y un punto de la constelación se obtiene la energía de la primera forma de onda S_1:

Equation:

$$\mathrm{Es}_1 = V^2\mathrm{Tb}$$

Para calcular la Energía de S_2 se hace exactamente el mismo procedimiento.

La introducción del ruido en el sistema ocasionará una situación como la descrita en la figura 3, es decir, en la constelación el punto correspondiente al símbolo no estará ubicado exactamente en el sitio que se ubicaría si no existiese el ruido. En las comunicaciones digitales se hace uso de un “valor umbral” con el cual el receptor distingue si el símbolo recibido es uno u otro. Este valor umbral es representado gráficamente en la constelación como una línea ubicada en el punto medio entre los puntos de los símbolos sin ruido.



Otro ejemplo de constelación, las líneas azules representan valores de umbral entre puntos cercanos

Autoevaluación

Exercise:

Problem:

¿Con qué base puede representarse un símbolo con un valor nulo en el intervalo [0, T]?

Solution:

Un valor nulo con dicha duración puede ser representado como la multiplicación de cualquier señal con la misma duración por cero, por lo cual se podrá representar con cualquier base, como por ejemplo, la base calculada por medio del otro símbolo. La posición del valor nulo en la constelación siempre será en el “origen”

Exercise:

Problem:

¿Cómo se observa en la constelación que hay un error en la transmisión?

Solution:

Hay un error en la transmisión si la representación de un símbolo se observa “más cerca” del punto correspondiente a otro símbolo diferente, es decir, si salta la línea de umbral.

Exercise:

Problem:

En base a la pregunta anterior, ¿Cómo puede reducirse la probabilidad de error?

Solution:

La probabilidad de error se reduce “aumentando la distancia” entre los puntos correspondientes a los símbolos, es decir, aumentando la potencia de transmisión (lo que aumenta el valor de V) o disminuyendo la velocidad de transmisión (lo que aumenta el valor de Tb).

Simuladores

ESTE VINCULO contiene una carpeta con un programa realizado en **LabVIEW** pero haciendo uso exclusivamente de "MATLAB Script" que calcula bases ortogonales por medio de Gram-Schmidt; puede obtenerse también un programa similar realizado netamente en LabVIEW por medio de ESTE VINCULO; la interfaz de ambos programas es prácticamente igual. Cada carpeta incluye el .vi correspondiente y todos los archivos necesarios para el funcionamiento de cada uno, si se elimina o renombra alguno de estos archivos, podría haber fallas en el funcionamiento del programa. La figura 6 contiene un video explicativo acerca del uso de los programas.

Ortogonalización Gram-Schmidt

[missing_resource: http://www.youtube.com/v/1MyIYtTuTXk?fs=1&hl=es_ES]

Video explicativo de la utilización del programa